

## A METHOD FOR SENSOR PLACEMENTS TAKING INTO ACCOUNT DIAGNOSABILITY CRITERIA

A. A. YASSINE, S. PLOIX, J. M. FLAUS

INPG/G-SCOP lab, BP 46, Saint Martin d'Herès 38402, France  
abed-alrahim.yassine@g-scop.inpg.fr, stephane.ploix@inpg.fr, jean-marie.flaus@inpg.fr

This paper presents a new approach for sensor placement based on diagnosability criteria. It is based on the study of structural matrices. Properties of structural matrices regarding detectability, discriminability and diagnosability are established in order to be used by sensor placement methods. The proposed approach manages any number of constraints modelled by linear or nonlinear equations and it does not require the design of analytical redundancy relations. Assuming that a constraint models a component and that the cost of the measurement of each variable is defined, a method determining sensor placements satisfying diagnosability specifications, where all the diagnosable, discriminable and detectable constraint sets are specified is proposed. An application example dealing with a dynamical linear system is presented.

**Keywords:** fault diagnosis, diagnosability, sensor placement, structural modelling

### 1. Introduction

In the scientific literature, many approaches of the fault diagnosis have been proposed since 1980. The FDI approach, which focuses on fault detection in dynamical systems, has been summarized in (Blanke, Kinnaert, Lunze and Staroswiecki, 2006). Related papers in this journal deal with the design of redundancy relations (Shumsky, 2007) and with the use of Fuzzy Logic (Dalton, Klotzke and Frank, 1999; Koscielny, Syfert and Bartys, 1999; Lopez-Toribio, Patton and Uppal, 1999) and neural networks (Korbicz, Patan and Obuchowicz, 1999; Witczak, 2006). The DX approach focuses on the diagnosis reasoning. It is summarized in (Hamscher, Console and De Kleer, 1992). Recently, a bridge approach between FDI and DX has been proposed (Cordier, Dague, Lévy, Dumas, Montmain, Staroswiecki and Travé-Massuyès, 2000; Nyberg and Krysander, 2003; Ploix, Touaf and Flaus, 2003). Thus, tools for solving diagnosis problems are now well established. However, designing an efficient diagnosis system does not start after the system design but it has to be done during the system design. Indeed, the performance of a diagnostic system highly depends on the number and on the location of actuators and sensors. Therefore, designing a system that has to be diagnosed not only require relevant fault diagnosis procedures but also efficient sensor placement algorithms.

(Madron and Veverka, 1992) has proposed a sensor placement method which deals with linear system. This method makes use of the Gauss-Jordan elimination to find a minimum set of variables to be measured. This ensures the observability of variables while simultaneously minimizing the cost of sensors. In this approach, the observable variables include the measurable variables plus the unmeasured but deductible variables. Another method for sensor placement has been proposed in (Maquin, Luong and Ragot, 1997). This method aims at guaranteeing the detectability and isolability of sensor failures. The proposed method is based on the concept of redundancy degree in variables and on the structural analysis of the system model. The sensor placement can be solved by an analysis of a cycle matrix or using the technique of mixed linear programming. (Commault, Dion and Yacoub Agha, 2006) has proposed an alternative method of sensor placement where a new set of separators (Irreducible Input Separators), which generates sets of system variables in which additional sensors must be implemented to solve the considered problem, is defined.

However, all these methods are not suitable for the design of systems that include a diagnosis system because, in this context, the goal of sensor placement should be to make it possible to monitor hazardous components. The sensor placement algorithm should compute solutions that satisfy detectability and diagnosability properties where

detectability is the possibility of detecting a fault on a component and diagnosability is the possibility of isolating a fault on a component without ambiguities with any other faulty components. Few methods have focused on this problem.

(Travé-Massuyès, Escobet and Milne, 2001) has proposed a method based on consecutive additions of sensors, which takes into account diagnosability criteria. The principle of this method is to analyze the physical model of a system from a structural point of view. This structural approach is based on Analytical Redundancy Relations (ARR) (Blanke *et al.*, 2006). However, this method requires an a priori design of all the ARR for a given set of sensors. Recently, (Frisk and Krysander, 2007) has proposed an efficient method based on a Dulmage-Mendelsohn decomposition (Dulmage and Mendelsohn, 1959; Pothén and Chin-Ju, 1990). Nevertheless, this method only applies to just-determined sets of constraints while most practical systems are under-determined when sensors are not taken into account, and over-determined afterwards.

This paper presents a new sensor placement algorithm that takes into account detectability and diagnosability specifications. It applies to systems for which only the structure is known. Thanks to this algorithm, sensor placements satisfying diagnosability objectives can be computed without designing all the ARRs, which is still an open problem. It applies to any system structurally depicted and does not assume just-determination. Section 2 details the main concepts that are useful to model systems for sensor placement. Then, section 3 presents how the sensor placement problem is formulated. Section 4 introduces tools for analyzing structural matrices. These tools are then used in section 5 to determine diagnosability properties directly from the analysis of structural matrices. Section 6 proposes basic algorithms for extracting blocks with useful properties from structural matrices and section 7 shows how to use these algorithms to compute sensor placements that satisfy diagnosability specifications. Finally, section 8 presents an application to an electronic circuit.

## 2. System modelling for sensor placement

Let us introduce the concepts and the formalism used in the paper to formalize the sensor placement problem. Behavioral knowledge starts with phenomena. A phenomenon is a potentially observable element of information about the actual state of a system. It is modeled by an implicitly time-varying variable, which has to be distinguished from a parameter that is model-dependent. Generally speaking, even if a phenomenon is observable, it is not possible to merge it with data because in fault diagnosis, data are only known providing that some actuators or sensors behave properly. Phenomena  $V(t) = \{\dots, v_i(t), \dots\}$  are linked to a phenomenological space

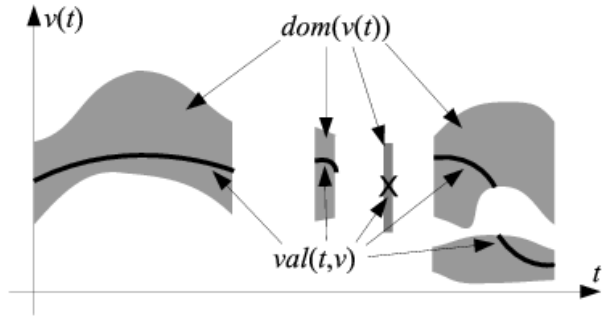


Fig. 1. A tube modeling a variable and an related observation

$\mathcal{F}(T, V) = \{V(t); t \in T\}$  where  $T$  stands for continuous or discrete time set. At any given time  $t$  in  $T$ , these phenomena belong to a domain  $dom(t, V) = dom(V(t))$  representing all the possible values that may have the phenomena. Consequently, when considering all  $t \in T$ ,  $\{dom(V(t)); t \in T\}$  represents a tube in the timed phenomenological space  $\mathcal{F}(T, V)$ .

All the phenomena have thus to be considered as unknown because observable phenomena are not observations. Let's introduce the concept of *data flow* to model actual data recorded on a system. A *data flow* models data provided by a source of information concerning a phenomenon. A data flow concerning a phenomenon  $v$  is denoted:  $val(t, v)$  with  $val(t, v) \in dom(t, v)$ . It corresponds to a trajectory belonging to the tube  $\{dom(t, v); t \in T\}$  (see figure 1). When information about  $v$  is coming from different sources, the different data flows can be denoted:  $val_i(t, v)$ . Formally, a data flow provided by a component  $c$  can be linked to a phenomenon:  $ok(c) \rightarrow \forall t \in T, val(t, v) = v$ , which means that if the component named  $c$  is in the mode  $ok$  then the data  $val(t, v)$  corresponds to the actual value of the phenomenon  $v$  at any time  $t \in T$ .

In fault diagnosis, a system is not supposed to remain in a given mode. Indeed, diagnostic analysis aims at retrieving the actual behavioral modes of the components of a system. At minimum, two modes are defined: the *ok* mode, which corresponds to the expected normal behavior and the *cf* mode, which is the complementary fault mode: it refers to all the behaviors that do not fit to the expected normal behavior. Sometimes, specific fault modes may be modeled (de Kleer and Williams, 1992; Struss, 1992). They are denoted by a specific label, for instance, the *leak* mode. Consider for instance a pipe where *ok* and *leak* are modeled. It yields:  $Modes(pipe) = \{ok, leak, cf\}$  where  $cf(pipe)$  refers to the behaviors that do not correspond to  $ok(pipe)$  or to  $leak(pipe)$ .

Except for the complementary fault mode, behavioral modes are modelled by cause-effect relationships between phenomena, which are represented by constraints. Each

constraint refers to a set of mappings containing unknown variables and known data flows. Generally speaking, a mapping over  $dom(t, V)$  is defined from one subspace  $dom(t, V_1)$  to another  $dom(t, V_2)$ , where  $\{V_1, V_2\}$  is a partition of  $V$ . Note that several mappings  $\kappa_i$  may model the same constraint  $k$ . If  $\kappa_i : dom(t, V_1) \mapsto dom(t, V_2)$  is a mapping representing a constraint  $k$  that models, for example, a component  $c_1$  in mode  $mode_1$  and a component  $c_2$  in mode  $mode_2$ , it yields:

$$mode_1(c_1) \wedge mode_2(c_2) \rightarrow V_2 = \kappa_i(t, V_1, val(V_3)); \quad (1)$$

$$V_1 \in dom(t, V_1), V_2 \in dom(t, V_2) \quad (2)$$

where the data flow  $val(V_3)$  is considered as included in the mapping.

But *constraint* is not strictly equivalent to *mapping*. A constraint corresponds to a set of equivalent mappings. Firstly, although mappings to multidimensional spaces could be used, they are difficult to manage. It is better to break them down into 1-dimensional mappings. In the following, 1-dimensional mappings modeling a constraint  $k$  are named *realizations* of  $k$ . Moreover, several realizations of a constraint may be equivalent. Let  $\kappa_i$  be a realization from  $V \setminus \{v\}$  to  $\{v\}$ . There may be equivalent realizations defined on  $V$  that also model the constraint. Therefore, the notion of *constraint* can be extended to represent all the equivalent realizations representing a given subset of  $dom(V)$ . In the following, a constraint  $k$  will be understood as a set of equivalent realizations. It is summarized by the set of variables occurring in the realizations:  $var(k)$ . It is assumed that if  $k$  is a constraint, for all  $v \in var(k)$ , there is an equivalent realisation  $\kappa_i : dom(t, var(k) \setminus \{v\}) \mapsto dom(t, v)$ .

To summarize, a system  $\Sigma$  is composed by a set of constraints  $K_\Sigma$  and a set of behavioral modes  $Modes(\Sigma)$  related to components in  $\Sigma$ .  $var(K_\Sigma)$  is the set of variables, named *port* in (Chittaro and Ranon, 2004), that models observable phenomena involved in  $\Sigma$ . Indeed, by extension, the set of variables appearing in a set of constraints  $K$  is denoted  $var(K) = \bigcup_{k \in K} var(k)$ . Each constraint  $\kappa \in K_\Sigma$  is linked to a mode  $m \in Modes(\Sigma)$  by a first order relationship:  $m \rightarrow \kappa$ . For the sake of simplicity, in this paper, it is considered that:

- only *ok* modes are considered in the sensor placement
- each constraint  $\kappa \in K_\Sigma$  models one mode and, conversely, that a mode can be modeled by at most one constraint.

The sensor placement problem consists then in defining the variables of  $var(\Sigma)$  that have to be measured to make it possible the detection and the identification of

*ok* modes from  $Modes(\Sigma)$ . These modes are denoted:  $Modes^{ok}(\Sigma)$ . From a mathematical point a view, it is a kind of combinatorial problem. Next section proposes a precise problem formulation.

### 3. Problem formulation

Let's present a intuitive formulation of the problem. Full definitions are given afterwards. The solving of a diagnostic problem is generally decomposed into two consecutive steps. The conflict or symptom generation, also called Fault Detection in Automatic Control community, and the diagnostic analysis also called Fault Isolation. The first step relies on consistency tests among minimal testable subsets of constraints<sup>1</sup>  $K \in K_\Sigma$  that include data flows (often called OBS for observations). Let  $\mathbb{K}$  be the set of minimal testable subsets of constraints. If  $K \in \mathbb{K}$  is a set of constraints leading to a test which is inconsistent, it means that, at least, one of the modes corresponding to the constraints of  $K$  is not actual. It is therefore important to trace the constraints belonging to a minimal testable subset  $K$  because it makes it possible the solving of the second sub-problem: the diagnostic analysis, which provides global conclusions in term of modes about the actual system states. The performance of a diagnostic system is highly dependent of the set  $\mathbb{K}$  and consequently, dependent of the set  $K_\Sigma$ , which highly dependent of the dataflows i.e. on the observations. Additional sensors lead to additional constraints in  $K_\Sigma$  and therefore, to new sets in  $\mathbb{K}$ .  $\mathbb{K}$  can be obtained from combinations of constraints from  $K_\Sigma$  using possible conflict generation (Pulido and Alonso, 2002), bipartite graph (Blanke *et al.*, 2006), Dulmage-Mendelsohn decomposition (Krysander, Aslund and Nyberg, 2005) or elimination rules (Ploix, Désinde and Touaf, 2005). Basically, once  $\mathbb{K}$  has been generated, it is possible to compute the performance of the diagnostic system in terms of detectability, discriminability or discernability, and diagnosability. Whether the performance satisfies the requested performances or not, the set  $K_\Sigma$  is modified and the process is done once again until requested performance are reached. However, this process requires lots of computations because the generation of  $\mathbb{K}$  is time consuming. Moreover, up to now, no one of these algorithms have been proved to be complete.

Another approach for sensor placement is proposed in this paper. It does not require the computation of  $\mathbb{K}$  from  $K_\Sigma$ . It directly solves the following problem by studying the structure of  $\Sigma$ . Let  $K_\Sigma$  be a set of constraints modeling the *ok* modes of a system  $\Sigma$ . Let  $var(K_\Sigma)$  be the variables appearing in  $K_\Sigma$ . The problem to be solved is: what are the complementary constraints modeling sensors dedicated to variables from  $var(K_\Sigma)$  that have to be

<sup>1</sup>minimal means that to be able to carry out a consistency test, no constraint can be removed from a subset

added to satisfy requested diagnosability performances.

Let's precise the problem formulation by defining the concept of testable subset or subsystem (TSS) of constraints and its relationship with the concept of ARR.

**Definition 1.** Let  $K$  be a set of constraints and  $v$  a variable in  $var(K)$  characterized by its domain  $dom(v)$ .  $K$  is a solving constraint set for  $v$  if using  $K$ , it is possible to instantiate  $v$  with a value set  $S$  such that  $S \subset dom(v)$ . A solving constraint set for  $v$  is minimal if there is no subset of  $K$ , which is also a solving constraint set for  $v$ . A minimal solving constraint set  $K$  for  $v$  is denoted:  $K \vdash v$ .

**Definition 2.** Let  $K$  be a set of constraints.  $K$  is testable if and only if there is a partition  $\{K_1, K_2\}$  of  $K$  and a variable  $v \in var(K)$  such that  $K_1 \vdash v$  and  $K_2 \vdash v$ . If this property is satisfied, it is indeed possible to check if the value set  $S_1$  deduced from  $K_1$  is consistent with the value set  $S_2$  deduced from  $K_2$ :  $S_1 \cap S_2 \neq \emptyset$ .

Adding any constraint to a testable set leads also to a testable set of constraints. Only minimal testable sets are interesting.

**Definition 3.** A testable set of constraints is minimal if it is not possible to keep testability when removing a constraint.

A global testable constraint that can be deduced from a TSS is called an analytical relation (ARR). Let  $\mathbb{K}_\Sigma = \{\dots, K_k, \dots\}$  be the set of all the testable subsystems that can be deduced from  $K_\Sigma$  according to (Blanke *et al.*, 2006; Ploix *et al.*, 2005). Because of the assumed one-to-one relationships between constraints and components, the notions of detectability and discriminability can be extended to constraints.

**Definition 4.** Let  $\mathbb{K}$  be a set of TSS coming from  $(K_\Sigma, C_\Sigma)$ . A constraint  $k \in K_\Sigma$  is detectable (Struss, Rehfus, Brignolo, Cascio, Console, Dague, Dubois, Dressler and Millet, 2002) in  $\mathbb{K}$  iff  $\exists K_i \in \mathbb{K} / k \in K_i$ . By extension, a set of constraints  $K \subset K_\Sigma$  is detectable in  $\mathbb{K}$  if  $\forall k_i \in K$ ,  $k_i$  is detectable in  $\mathbb{K}$ .

**Definition 5.** Two constraints  $(k_1, k_2) \in K_\Sigma^2$  are discriminable (Struss *et al.*, 2002) in  $\mathbb{K}$  if:  $\exists K_i \in \mathbb{K} / k_1 \in K_i$  and  $k_2 \notin K_i$  or if  $\exists K_j \in \mathbb{K} / k_2 \in K_j$  and  $k_1 \notin K_j$ . By extension, the constraints of a set  $K \subset K_\Sigma$  are discriminable in  $\mathbb{K}$  iff:  $\forall (k_i, k_j) \in K^2$ ,  $k_i$  and  $k_j$  are discriminable in  $\mathbb{K}$  with  $k_i \neq k_j$ .

Obviously, non detectability implies non discriminability.

**Definition 6.** A constraint  $k \in K_\Sigma$  is diagnosable (Struss *et al.*, 2002; Console, Picardi and Ribando, 2000) in  $\mathbb{K}$  iff: it is detectable and  $\forall k_j \in (K_\Sigma \setminus k)$ ,  $(k, k_j)$  are discriminable in  $\mathbb{K}$ . By extension, constraints  $K \subset K_\Sigma$  are diagnosable in  $\mathbb{K}$  iff:  $\forall k_i \in K$ ,  $k_i$  are diagnosable in  $\mathbb{K}$ .

In order to formulate the sensor placement problem, the notion of terminal constraint has to be introduced.

**Definition 7.** A terminal constraint  $k$  is a constraint that satisfies:  $card(var(k)) = 1$  where  $var(k)$  is the set of variables appearing in the constraint  $k$ .

A terminal constraint usually models a sensor or an actuator. It is thus a major concept in sensor placement. Note that if a candidate sensor measures not only one variable  $v$  but a combination of several variables  $v_1, \dots, v_n$ , a new constraint  $k$  satisfying  $var(k) = \{v_1, \dots, v_n, v_*\}$ , where  $v_*$  is a virtual measurable variable, has to be added into  $K_\Sigma$ . Then, the solving is similar to standard problem.

In fault diagnosis, sensor placement has to satisfy specifications dealing with detectability and diagnosability. Because a one-to-one relation between components and constraints is assumed, what is true for components is also true for constraints. In the following, only constraints will be considered: the analogy with components is implicit. In this paper, complete specifications are considered. Partial specifications can also be managed: they will be presented in a next paper. These complete specifications consist in a partition of the constraint set  $K_\Sigma$  into the following subsets:

- the set of constraints  $K_{diag}$  that must be diagnosable
- the set of subsets of constraints  $\mathbb{K}_{nondis} = \{\dots, K_i, \dots\}$  for which each set  $K_i$  must be non discriminable but detectable
- the set of constraints  $K_{nondet}$  that must be non detectable.

Complete specifications  $K_{diag}$ ,  $\mathbb{K}_{nondis}$  and  $K_{nondet}$  for sensor placement problems are meaningful if the two following properties are satisfied:

1. Sets in specifications must not overlap one each other to make sense. Constraint sets have to satisfy:  $K_{nondet} \cap K_{diag} = \phi$ ,  $\forall K_i \in \mathbb{K}_{nondis}$ ,  $K_i \cap K_{nondet} = \phi$ ,  $\forall K_i \in \mathbb{K}_{nondis}$ ,  $K_i \cap K_{diag} = \phi$  and  $\forall (K_i, K_j) \in \mathbb{K}_{nondis}^2$ ,  $K_i \cap K_j = \phi$  if  $K_i \neq K_j$  (no overlapping property).
2. The union of all the components appearing in  $K_{diag}$ ,  $\mathbb{K}_{nondis}$  and  $K_{nondet}$  has to correspond to  $K_\Sigma$ :  $K_\Sigma = K_{diag} \cup K_{nondet} \cup \bigcup_{K_i \in \mathbb{K}_{nondis}} K_i$  (completeness property).

If these properties are satisfied the complete specifications are qualified as consistent in  $K_\Sigma$ .

Satisfying the diagnosability specifications requires information delivered by sensors. Let  $K'_\Sigma$  represent the system  $\Sigma$  with additional sensors where  $K_{\Sigma'}$  contains the constraints  $K_\Sigma$  of the system  $\Sigma$  plus the additional terminal constraints modeling the additional sensors. Therefore, solving a sensor placement problem consists in determining the additional terminal constraints in  $K_{\Sigma'}$  that lead to the satisfaction of complete specifications.

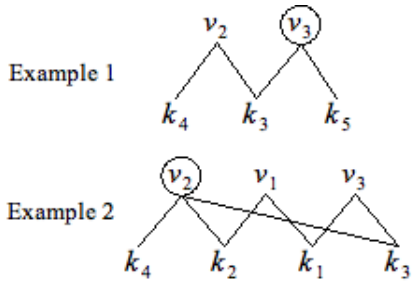


Fig. 2. Link between propagations and minimal testable subsets

In the next sections, diagnosability properties of structural matrices are established and used for the design on sensor placement satisfying diagnosability specifications.

#### 4. Basic properties of structural matrices

Before pointing out diagnosability properties, some basic properties of structural matrices have to be established. The constraints  $K_\Sigma = \{\dots, k_i, \dots\}$  can be represented by a structural matrix  $\mathcal{M}_\Sigma$ , which is an incidence matrix representing the application  $\mathcal{M}_\Sigma : var(K_\Sigma) \rightarrow K_\Sigma$ .

According to the definition, an TSS is a minimum set of constraints  $K$  such that there is at least one variable for which two different minimal solving sets can be found. A minimal solving set leading to a variable  $v$  corresponds to a value propagation (Apt, 2003) starting usually, but not necessary, by terminal constraints and leading to  $v$ . Therefore, a TSS can also be seen as two distinct value propagations leading to a given variable. This point of view has been adopted as a theoretical tool to develop proofs.

Let  $k_1$  and  $k_2$  be two constraints. The propagation of a variable  $v$  between  $k_1$  and  $k_2$  is possible only if  $v \in var(k_1) \cap var(k_2)$ . The variable  $v$  is qualified as propagable between  $k_1$  and  $k_2$ :  $v$  is a link between  $k_1$  and  $k_2$ . In the corresponding structural matrix, this link is represented by a thick line:

	$v$	...
$k_1$	<b>1</b>	...
$k_2$	<b>1</b>	...

Consider now a system, defined by  $K_\Sigma = \{k_1, k_2, k_3, k_4, k_5\}$  with  $var(k_1) = \{v_1, v_3\}$ ,  $var(k_2) = \{v_1, v_2\}$ ,  $var(k_3) = \{v_2, v_3\}$ ,  $var(k_4) = \{v_2\}$  and  $var(k_5) = \{v_3\}$ . Terminal constraints  $k_4$  and  $k_5$  model sensors or actuators. Each terminal constraint contains known data. Figure 2 represents examples of propagations that lead to TSS with a bipartite graph. But in a bipartite graph, links do not appear clearly: they correspond to alternate paths (or chains) with this pattern: constraint-

variable-constraint. Links appear more clearly in structural matrices as lines linking two constraints. In the following structural matrices, the variables surrounded by a circle represent the variables that can be instantiated twice. The relevance of links remains obvious in example 2, where a propagation do not start by a terminal constraint. The paths corresponding to propagations of solving sets have been drawn. Variable  $v_2$  has been instantiated twice.

	$v_1$	$v_2$	$v_3$
$k_1$	1		1
$k_2$	1	1	
$k_3$		1	1
$k_4$		1	
$k_5$			1

	$v_1$	$v_2$	$v_3$
$k_1$	1		1
$k_2$	1	1	
$k_3$		1	1
$k_4$		1	
$k_5$			1

Once again, paths may be reduced to links (thick lines). The following example points out another structural matrix with two propagations leading to variable  $v_3$ .

	$v_1$	$v_2$	$v_3$
$k_1$	1	1	1
$k_2$	1	1	
$k_3$		1	
$k_4$			1

The concept of linked constraints has to be formalized because discriminability depends on this concept. Before defining *linked constraints*, the concept of *inter-connected constraints* has to be introduced. The constraints of a system  $\Sigma$  may be modeled by a non directed bipartite graph  $(K_\Sigma, var(K_\Sigma), E_\Sigma)$  where  $E_\Sigma$  is the set of edges. Each edge  $e = (k, v)$  models that  $v \in var(k)$ . **Definition 8.** A set of constraints  $K \subset K_\Sigma$  is inter-connected by a set of variables  $V \subset var(K_\Sigma)$  iff there is a tree  $(K, V, E) \subset (K_\Sigma, var(K_\Sigma), E_\Sigma)$  with constraints at extremities (see (Bollobás, 1998) for example), which satisfies  $card(V) = card(K) - 1$ .

To point out the link with bipartite graph theory, if  $K$  is interconnected by  $V$  in  $K_\Sigma$ ,  $V$  is necessarily a complete coupling for  $K$  with respect to variables. The notion of *linked set of constraints* can now be introduced.

**Definition 9.** A set of constraints  $K \subset K_\Sigma$  is linked in  $K_\Sigma$  by a set of variables  $V \subset var(K_\Sigma)$  iff  $K$  is inter-connected by  $V$  and iff the other constraints of  $K_\Sigma$  (i.e.  $K_\Sigma \setminus K$ ) do not contain any variable of  $V$ . The variables of  $V$  are called linking variables for  $K$ . They are denoted:  $var_{linking}(K, K_\Sigma)$ .

The shape of a structural matrix dealing with linked constraints is drawn in figure 3.

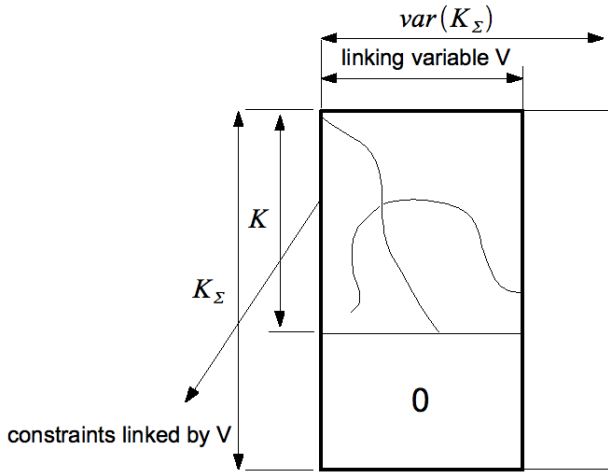


Fig. 3. Structural matrix of a constraint set  $K$ , which is linked by a set of variables  $V$

The concept of *linked constraints* is strongly connected with discriminability.

**Lemma 1** A set of constraints  $K \subset K_\Sigma$  linked by a set of variables  $V \subset V_\Sigma$  is necessarily non discriminable.

**Proof 1** Indeed,

1. because variables in  $V$  only appear in the constraints belonging to  $K$ , the only way for propagating variables is to use the constraints in  $K$  and the variables in  $V$
2. because there is a tree  $(K, V, E) \subset (K_\Sigma, var(K_\Sigma), E_\Sigma)$  with constraints at extremities, instantiating all the variables in  $V$  involves at least the achievement of the propagations defined by the tree.

Therefore, all the constraints are invariably found together in TSS. In order to improve the clarity of these explanations, let us introduce the notion of stump variables.

**Definition 10.** A set of variables  $var(K)$  appearing in a set of constraints  $K$  but not in the other constraints of  $K_\Sigma$  (i.e.  $K_\Sigma \setminus K$ ) are named stump variables in  $K_\Sigma$  with respect to  $K$ . They are denoted:  $var_{stump}(K, K_\Sigma)$ .

For instance, the set of variables  $V$  that links a set of constraints  $K$  belong to the stump variables  $var_{stump}(K, K_\Sigma)$  with  $K \subset K_\Sigma$ .

A set of constraints cannot be used to generate a TSS if they are linked and if there are additional variables that cannot be propagated. These constraints are qualified as isolated. Detectability depends on this concept.

**Definition 11.** A set of several constraints  $K \subset K_\Sigma$  is isolated in  $K_\Sigma$  by a set of variables  $V \subset var(K_\Sigma)$  if it is linked by  $V$  and if there is at least one variable in

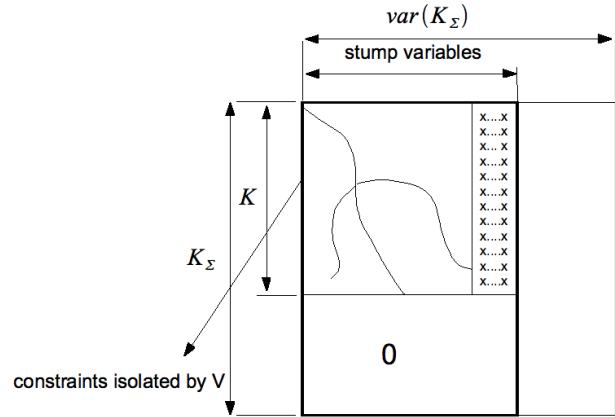


Fig. 4. Structural matrix of a constraint set, which is isolated by the set of variables  $V$

$var(K) \setminus V$  that does not belong to other constraints of  $K_\Sigma$  (i.e.  $K_\Sigma \setminus K$ ). If the set contains only one constraint, the link condition disappears.

The shape of a structural matrix dealing with isolated constraints is drawn in figure 4.

The concept of isolated constraints is strongly linked with detectability.

**Lemma 2** A set of constraints  $K \subset K_\Sigma$  isolated in  $K_\Sigma$  by  $V$  is necessarily non detectable.

**Proof 2** The constraints  $K$  isolated in  $K_\Sigma$  by  $V$ , will always come together in TSS because, by definition, they are linked by  $V$ . Because of the fact that, in isolated constraints, there is at least one additional variable in  $var(K)$  which does not appear in other constraints (i.e.  $K_\Sigma \setminus K$ ), it is not possible to instantiate this variable and, therefore, this set of constraints cannot be involved into a TSS: constraints  $K$  are thus non detectable.

## 5. Diagnosability properties of structural matrices

This section aims at setting up a direct link from sets of constraints to detectability and diagnosability properties. Firstly, it is obvious that adding additional constraints connected to all the variables  $var(k)$  appearing in a constraint  $k$ , ensures the diagnosability of  $k$ .

**Lemma 3** Let  $k \in K_\Sigma$  be a constraint. If additional terminal constraints dealing with all the variables in  $var(k)$  are added, then the constraint  $k$  is diagnosable.

**Proof 3** Because there are additional terminal constraints connected to each variable in  $V(k)$ , a value can be assigned for each variable. Consequently, there is one TSS containing  $k$  plus additional terminal constraints

connected to variables in  $\text{var}(k)$ . Therefore, the constraint  $k \in K_\Sigma$  is necessarily diagnosable because there is one TSS that does not contain other constraints of  $K_\Sigma$  (i.e.  $K_\Sigma \setminus \{k\}$ ).

Lemma 3 can be directly applied to all the constraints of a constraint set.

**Corollary 1** *If additional terminal constraints dealing with all the variables  $\text{var}(K)$  of a constraint set  $K \in K_\Sigma$ , then each constraint  $k \in K$  is diagnosable.*

In lemma 2, a relationship between isolated constraints and the detectability property has been presented. The next lemma generalizes the previous results.

**Lemma 4** *A sufficient condition for a subset of constraints  $K \subset K_\Sigma$  to be non detectable is that there is a tuple  $(K_1, \dots, K_m)$  of  $m$  sets of constraints making up a partition  $\mathcal{P}(K)$  of  $K$  such that each  $K_i$  is isolated in  $K_\Sigma \setminus \bigcup_{j < i} K_j$  ( $K_1$  is a limit case: it should be isolated in  $K_\Sigma$ ).*

**Proof 4** *The case of  $K_1$  has been discussed in lemma 2: because the constraints in  $K_1$  are isolated in  $K_\Sigma$ , they are non detectable and therefore cannot be included in TSS. Then, the remaining candidate constraints for TSS belong to  $K_\Sigma \setminus K_1$ . Because  $K_2$  is isolated in  $K_\Sigma \setminus K_1$ , they are non detectable. The reasoning can be extended to any  $K_i$ . Consequently, the constraints in  $K = \bigcup_i K_i$  are non detectable.*

Figure 5 indicates the shape of a structural matrix of non detectable constraints.

Consider, for example, a system modeled by the following structural matrix:

	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_6$
$k_1$	1	0	0	1	0	0
$k_2$	0	1	1	0	1	0
$k_3$	0	1	1	0	1	0
$k_4$	0	0	0	1	0	1
$k_5$	0	0	0	1	1	1

Assume that the set  $K = \{k_1, k_2, k_3\}$  is required to be non detectable. In this example, it exists a tuple  $(\{k_1\}, \{k_2, k_3\})$  such as each element  $K_i$  satisfies lemma 4. If there are no additional terminal constraints containing  $v_1, v_2$  and  $v_3$ , the subset  $K$  is necessary non detectable.

**Lemma 5** *A sufficient condition for each set  $K_i \subset K$  belonging to a set of  $m$  constraint sets  $\mathbb{K} = \{K_1, \dots, K_m\}$  such that  $\forall K_i \neq K_j, K_i \cap K_j = \emptyset$ , to be non discriminable is that each  $K_i$  is linked by a set of variables  $V_i$ .*

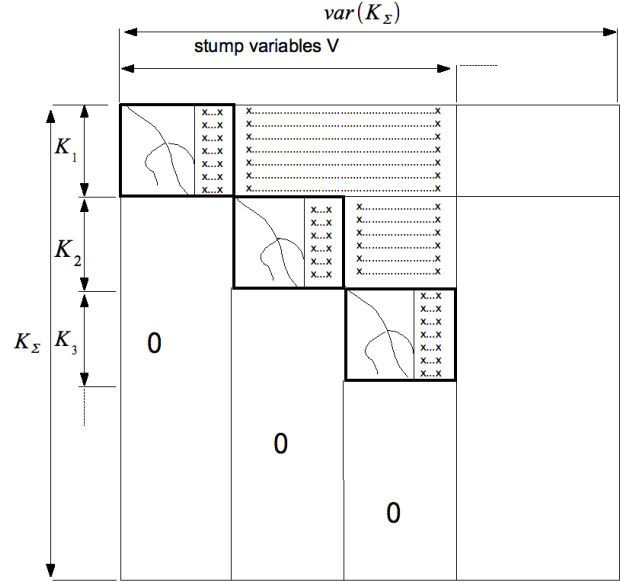


Fig. 5. Structural matrix of non detectable constraints

**Proof 5** *This lemma is a direct application of lemma 1 to several sets of constraints.*

Consider, for example, a system modeled by the following structural matrix:

	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$
$k_1$	1	0	1	1	1
$k_2$	1	1	1	1	0
$k_3$	1	1	1	0	1
$k_4$	0	1	1	0	0
$k_5$	0	0	0	1	1

Assume that  $K = \{k_1, k_2, k_3, k_4\}$  is a constraint subset that should be non discriminable. Because the constraints  $k_1, k_2, k_3$  and  $k_4$  are linked by  $V = \{v_1, v_2, v_3\}$ , lemma 5 is satisfied. Therefore,  $k_1, k_2, k_3$  and  $k_4$  are non discriminable provided that no additional terminal constraints contain a variable of  $V$ .

The following theorem gathers lemmas 3, 4 and 5.

**Theorem 1** *Let  $K_\Sigma$  be a set of constraints and  $K_{nondet}$ ,  $\mathbb{K}_{nondis}$  and  $K_{diag}$  be the specifications of a sensor placement problem consistent in  $K_\Sigma$ . Sufficient conditions for the specifications to be fulfilled are:*

1. *there exists a tuple  $(K_1, \dots, K_p)$  of  $p$  sets of constraints making up a partition  $\mathcal{P}(K_{nondet})$  of  $K_{nondet}$  such that each  $K_i$  is isolated in  $K_\Sigma \setminus \bigcup_{j < i} K_j$  ( $K_1$  is a limit case: it should be isolated in  $K_\Sigma$ ) as shown in figure 5.*
2. *each set  $K_i$  belonging to  $\mathbb{K}_{nondis} = \{K_1, \dots, K_m\}$  such that  $\forall K_i \neq K_j, K_i \cap K_j = \emptyset$ , is linked by a*

set of variables  $V_i$  in considering only the detectable constraints  $K_\Sigma \setminus K_{nondet}$

3. Additional terminal constraints are added on the variables  $V_{candidate} = \text{var}(K_\Sigma) \setminus (\text{var}_{stump}(K_{nondet}, K_\Sigma) \cup \bigcup_{K_j \in \mathbb{K}_{nondis}} \text{var}_{linking}(K_j, K_\Sigma \setminus K_{nondet}))$ .

**Proof 6** The proof relies on the resulting structure of the structural matrix, which directly stems from corollary 1 and lemmas 4 and 5. Note that point 2 could also be stated for the whole set of constraints  $K_\Sigma$ . However, it is not useful to include non detectable constraints, which will not appear in resulting TSS: it would be less conservative.

Because of lemma 4 and 5, the variables of  $\text{var}(K_{diag})$  cannot contain variables appearing in the variables involved in (1) and (2) i.e. in  $\text{var}_{stump}(K_{nondet}, K_\Sigma)$  and in  $\bigcup_{K_j \in \mathbb{K}_{nondis}} \text{var}_{linking}(K_j, K_\Sigma \setminus K_{nondet})$ . Then,  $\text{var}(K_{diag})$  satisfies:  $\text{var}(K_{diag}) \subset V_{candidate}$ . Because the variables of  $V_{candidate}$  can be instantiated with measured values, all the constraints of  $K_{diag}$  are diagnosable following corollary 1.

The point which has to be proved is that, in specifications,  $\mathbb{K}_{nondis}$  defines non discriminable but detectable sets and not only non discriminable sets as in lemma 5: the detectability of sets in  $\mathbb{K}_{nondis}$  has to be proved.

The variables  $\text{var}(K_i)$  of a constraint set  $K_i \in \mathbb{K}_{nondis}$  can be decomposed into two sets:  $V_i^-$  and  $V_i^+$  where  $V_i^- = \text{var}_{linking}(K_i, K_\Sigma \setminus K_{nondet})$  contains the linking variables and  $V_i^+$  contains the remaining variables  $V_i^+ = \text{var}(K_i) \setminus V_i^-$ . Because of lemma 4 and 5, the set  $V_i^+$  cannot contain variables in  $\text{var}_{stump}(K_{nondet}, K_\Sigma)$  and in  $\bigcup_{K_j \in \mathbb{K}_{nondis}; K_j \neq K_i} \text{var}_{linking}(K_j, K_\Sigma)$ . Therefore,  $V_i^+$  satisfies:  $V_i^+ \subset V_{candidate}$ .

Because of the third point of the theorem, all the variables of  $V_{candidate}$  are known: additional terminal constraints are indeed added, there is necessarily a TSS dealing with all the constraints in  $K_i$ . It proves that the constraint set  $K_i$  is necessarily detectable. Because this result holds for any  $K_i \in \mathbb{K}_{nondis}$ , it proves the theorem.

Satisfying theorem 1 guarantees that the specifications are satisfied. However, because the theorem provides only a sufficient condition for diagnosability, the number of additional terminal constraints is not necessarily minimal. It has to be checked afterwards.

In the next section, an algorithm for extracting blocks from a structural matrix is presented. This algorithm is required by methods for sensor placement based on complete specifications.

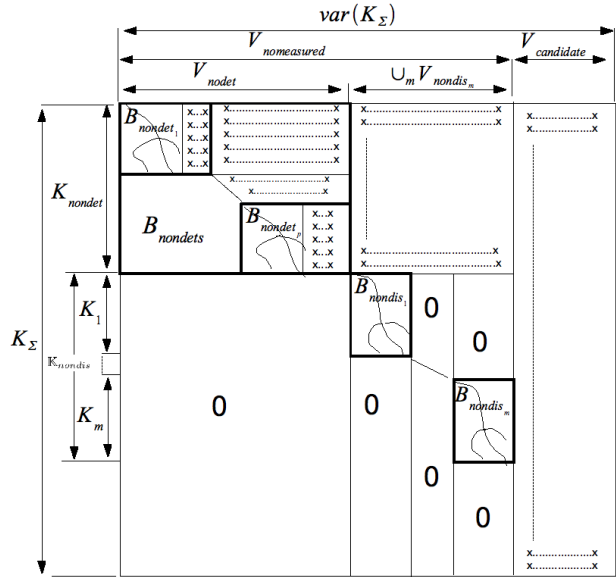


Fig. 6. Shape of a structural matrix satisfying theorem 1

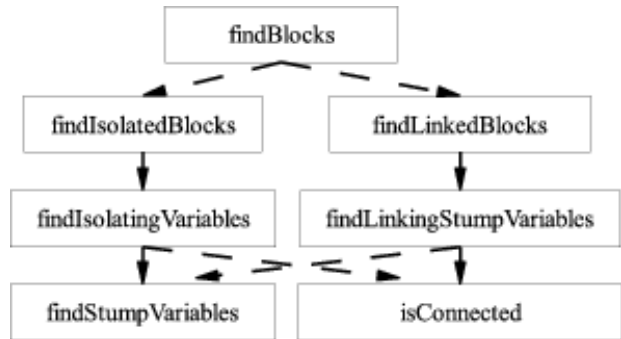


Fig. 7. Dependency scheme of the block extraction algorithm

## 6. Extracting blocks from a structural matrix

Before presenting an algorithm for extracting blocks from a structural matrix  $K_\Sigma$ , let's introduce some notations. Firstly, the notion of block is formalized: a *block* is a couple defined by  $block = (K, V)$  where  $block.cons = K$  and  $block.var = V$  stand respectively for a set of constraints and a set of variables. Two blocks can be merged:

$$\text{merge}(block_1, block_2) = \text{block}(block_1.cons \cup block_2.cons, block_1.var \cup block_2.var)$$

A set of blocks is denoted with the symbol  $\mathbb{B}$ . By extension, the block resulting from the merging of sets of blocks  $\mathbb{B}$  is denoted:  $\text{merge}(\mathbb{B})$ .

Figure 7 represents the dependency scheme between the methods that are defined. The main algorithm is

named `findBlocks` (algorithm 1). It extracts the different blocks that appear in theorem 1, considering only the variables  $V_\Delta$ .

---

**Algorithm 1** `findBlocks( $K_\Sigma, V_\Delta$ )`: A tuple of block sets ( $\mathbb{B}_{nondet}, \mathbb{B}_{nondis}, B_{diag}$ ), considering only the variables  $V_\Delta$

---

**Require:**  $V_\Delta \subseteq var(K_\Sigma)$   
 $K_\Delta \leftarrow K_\Sigma$   
 $\mathbb{B}_{nondet} \leftarrow \text{findIsolatedBlocks}(K_\Sigma, K_\Delta, V_\Delta)$   
 $K_\Delta \leftarrow K_\Delta \setminus \text{merge}(\mathbb{B}_{nondet}).cons$   
 $\mathbb{B}_{nondis} \leftarrow \text{findLinkedBlocks}(K_\Sigma \setminus \text{merge}(\mathbb{B}_{nondet}).cons, K_\Delta, V_\Delta \setminus \text{merge}(\mathbb{B}_{nondet}).var)$   
 $K_{diag} \leftarrow K_\Delta \setminus \text{merge}(\mathbb{B}_{nondis}).cons$   
**return** ( $\mathbb{B}_{nondet}, \mathbb{B}_{nondis}, \text{block}(K_{diag}, var(K_{diag}))$ )

---

In order to describe methods `findIsolatedBlocks()` and `findLinkedBlocks()`, the notions of  $Knode$  and  $buffer$  of  $Knodes$  are introduced. A  $Knode$  is a couple of constraint sets:  $Knode = Knode(K^-, K^+)$ , where  $Knode^- = K^-$  and  $Knode^+ = K^+$ . A  $buffer$  is a special First In First Out buffer. The basic functionalities are:  $buffer.push(Knode)$  and  $buffer.pop()$ . They respectively correspond to add a  $Knode$  in the buffer and get a  $Knode$  from the buffer.

Using these notions, the algorithm `findIsolatedBlocks()` (algorithm 2) extracts the set of isolated blocks from a set of constraints  $K_\Delta \subseteq K_\Sigma$ , considering only the variables  $V_\Delta$ . According to lemma 4, the constraints belonging to the resulting blocks are not detectable.

This algorithm depends on the `findIsolatingVariables()` method. It is given by algorithm 3.

The algorithm `findLinkedBlocks()` (algorithm 4) extracts the set of linked constraints from a set  $K_\Delta \subseteq K_\Sigma$ , considering only the variables  $V_\Delta$ . The structure of this algorithm is very closed to the structure of algorithm 2. According to lemma 5, the constraints belonging to the resulting blocks are not discriminable.

This algorithm depends on the `findLinkingStumpVariables()` method, which is given by algorithm 5.

Finally, according the figure 7, the two algorithms `findIsolatingVariables()` and `findLinkedBlocks()` depends on two methods `findStumpVariables()` (algorithm 6) and `isInterconnected()` (algorithm 7).

The top-level method `findBlocks( $K_\Sigma$ )` leads to the blocks depicted in figure 6. These results are very useful to support the sensor placement. Indeed, constraints belonging to  $B_{diag}.cons$  are already diagnosable. Therefore, finding a sensor placement satisfying specifications

---

**Algorithm 2** `findIsolatedBlocks( $K_\Sigma, K_\Delta, V_\Delta$ )`: A block containing the set of isolated constraints subsets of  $K_\Delta$  in  $K_\Sigma$  and the isolating variables, considering only the varcostiables  $V_\Delta$

---

**Require:**  $K_\Delta \subseteq K_\Sigma$   
**Require:** A  $buffer$  is created  
 $buffer \leftarrow \emptyset$   
 $\mathbb{B} \leftarrow \emptyset$  {an empty list of blocks}  
 $buffer.push(Knode(\emptyset, K_\Delta))$   
**while**  $buffer \neq \emptyset$  **do**  
 $Knode \leftarrow buffer.pop()$   
 $K \leftarrow K_\Delta \setminus Knode^-$   
 $V \leftarrow \text{findIsolatingVariables}(K_\Sigma, K, V_\Delta)$   
**if**  $V \neq \emptyset$  **then**  
 $\mathbb{B} \leftarrow (\mathbb{B}, \text{block}(Knode^+, V))$   
 $K_\Delta \leftarrow Knode^-$   
 $K_\Sigma \leftarrow K_\Sigma \setminus K$   
 $V_\Delta \leftarrow V_\Delta \setminus V$   
 $buffer \leftarrow \emptyset$   
 $buffer.push(Knode(\emptyset, K_\Delta))$   
**else**  
 $K^+ \leftarrow Knode^+$   
**for all**  $k \in Knode^+$  **do**  
 $K^- \leftarrow Knode^- \cup \{k\}$   
 $K^+ \leftarrow K^+ \setminus \{k\}$   
**if**  $K^- \neq K_\Delta$  **then**  
 $buffer.push(Knode(K^-, K^+))$   
**end if**  
**end for**  
**end if**  
**end while**  
**return**  $\mathbb{B}$

---

requires that the specified  $K_{diag}^{spec}$  includes  $B_{diag}.cons$ :

$$B_{diag}.cons \subset K_{diag}^{spec} \quad (3)$$

In the same way, the constraints  $\text{merge}(\mathbb{B}_{nondis}).cons \cup B_{diag}.cons$  are already detectable. Therefore, finding a sensor placement satisfying specifications requires that the specified  $K_{nondet}^{spec}$  is included in  $\text{merge}(\mathbb{B}_{nondet}).cons$ :

$$K_{nondet}^{spec} \subset \text{merge}(\mathbb{B}_{nondet}).cons \quad (4)$$

## 7. Method for sensor placement

A method for optimal sensor placements satisfying diagnosability specifications is proposed in this section. This method deals with complete specifications:  $K_{diag}^{spec}$ ,  $\mathbb{K}_{nondis}^{spec}$  and  $K_{nondet}^{spec}$  (see section 4).

There may be several sensor placements that satisfy diagnosability specifications. In order to select the most interesting one, a criterion based on the cost of the sensor placement is considered. Let's introduce the following notations. The cost of the measurement of a variable

---

**Algorithm 3** findIsolatingVariables( $K_\Sigma, K_\Delta, V_\Delta$ ): A set of variables isolating  $K_\Delta$  in  $K_\Sigma$ , considering only the variables  $V_\Delta$

---

**Require:**  $K_\Delta \subseteq K_\Sigma$   
 $V_{stump} \leftarrow \text{findStumpVariables}(K_\Sigma, K_\Delta, V_\Delta)$   
**if**  $\text{card}(V_{stump}) \geq \text{card}(K_\Delta)$  **then**  
   **if**  $\text{card}(K_\Delta) = 1$  **then**  
     **return**  $V_{stump}$   
**else**  
   **for**  $V \in \text{combinations of } \text{card}(K_\Delta) - 1 \text{ variables from } V_{stump}$  **do**  
     **if**  $\text{isInterconnected}(K_\Delta, V)$  **then**  
       **return**  $V_{stump}$   
     **end if**  
**end for**  
**end if**  
**end if**  
**return**  $\emptyset$

---



---

**Algorithm 4** findLinkedBlocks( $K_\Sigma, K_\Delta, V_\Delta$ ): A set of blocks, where each one corresponds to a linked but not isolated set of constraints, and its corresponding linking variables, considering only the variables  $V_\Delta$

---

**Require:**  $K_\Delta \subseteq K_\Sigma$   
**Require:** A *buffer* is created  
 $\text{buffer} \leftarrow \emptyset$   
 $\mathbb{B} \leftarrow \emptyset$  {an empty list of blocks}  
 $\text{buffer}.\text{push}(Knode(\emptyset, K_\Delta))$   
**while**  $\text{buffer} \neq \emptyset$  **do**  
    $Knode \leftarrow \text{buffer}.\text{pop}()$   
    $K \leftarrow K_\Delta \setminus Knode^-$   
    $V \leftarrow \text{findStumpLinkingVariables}(K_\Sigma, K, V_\Delta)$   
   **if**  $V \neq \emptyset$  **then**  
      $\mathbb{B} \leftarrow (\mathbb{B}, \text{block}(Knode^+, V))$   
      $K_\Delta \leftarrow Knode^-$   
      $\text{buffer} \leftarrow \emptyset$   
      $\text{buffer}.\text{push}(Knode(\emptyset, K_\Delta))$   
**else**  
    $K^+ \leftarrow Knode^+$   
   **for all**  $k \in Knode^+$  **do**  
      $K^- \leftarrow Knode^- \cup \{k\}$   
      $K^+ \leftarrow K^+ \setminus \{k\}$   
     **if**  $K^- \neq K_\Delta$  **then**  
        $\text{buffer}.\text{push}(Knode(K^-, K^+))$   
     **end if**  
**end for**  
**end if**  
**end while**  
**return**  $\mathbb{B}$

---



---

**Algorithm 5** findLinkingStumpVariables( $K_\Sigma, K_\Delta, V_\Delta$ ): One set of stump variables linking  $K_\Delta$  in  $K_\Sigma$ , considering only the variables  $V_\Delta$

---

**Require:**  $K_\Delta \subseteq K_\Sigma$   
 $V_{stump} \leftarrow \text{findStumpVariables}(K_\Sigma, K_\Delta, V_\Delta)$   
**for**  $V \in \text{combinations of } \text{card}(K_\Delta) - 1 \text{ variables from } V_{stump}$  **do**  
   **if**  $\text{isInterconnected}(K_\Delta, V)$  **then**  
     **return**  $V_{\text{linkingStump}}$   
**end if**  
**end for**  
**return**  $\emptyset$

---



---

**Algorithm 6** findStumpVariables( $K_\Sigma, K_\Delta, V_\Delta$ ): A set of stump variables for  $K_\Delta$  in  $K_\Sigma$ , considering only the variables  $V_\Delta$

---

**Require:**  $K_\Delta \subseteq K_\Sigma$   
 $V_{stump} \leftarrow \emptyset$   
 $V_{\text{nonstump}} \leftarrow \emptyset$   
**for all**  $v \in V_\Delta$  **do**  
   **if**  $\text{cons}(K_\Sigma, v) \subset K_\Delta$  **then**  
      $V_{stump} \leftarrow V_{stump} \cup \{v\}$   
**else**  
      $V_{\text{nonstump}} \leftarrow V_{\text{nonstump}} \cup \{v\}$   
**end if**  
**end for**  
**return**  $V_{stump}$

---



---

**Algorithm 7** isInterconnected( $K_\Delta, V$ ): True if constraints  $K_\Delta$  are interconnected by  $V$

---

**Require:**  $K_\Delta \subseteq K_\Sigma$   
**Require:** An empty *buffer* is created  
**if**  $V \subseteq \text{var}(K_\Delta) \wedge \text{card}(V) = \text{card}(K_\Delta) - 1$  **then**  
    $\text{buffer}.\text{push}(Vnode(\emptyset, V))$   
   **while**  $\text{buffer} \neq \emptyset$  **do**  
      $Vnode \leftarrow \text{buffer}.\text{pop}()$   
      $V^+ \leftarrow Vnode^+$   
     **for all**  $v \in Vnode^+$  **do**  
        $V^- \leftarrow Vnode^- \cup \{v\}$   
        $K^+ \leftarrow \text{cons}(V^-)$   
       **if**  $\text{card}(K^+) = \text{card}(V^-)$  **then**  
         **return** false  
       **else**  
          $V^+ \leftarrow V^+ \setminus \{v\}$   
         **if**  $V^+ \neq \emptyset$  **then**  
            $\text{buffer}.\text{push}(Knode(V^-, V^+))$   
         **end if**  
       **end if**  
     **end for**  
   **end while**  
   **return** true  
**else**  
   **return** false  
**end if**

---

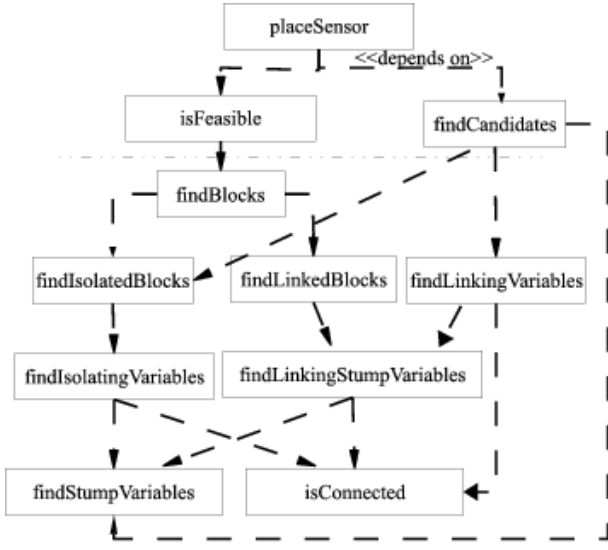


Fig. 8. Dependency scheme of the sensor placement method for complete specifications

$v$  is denoted  $cost(v)$ . By extension, the cost of the measurement of a set of variables  $V$  is denoted:  $cost(V) = \sum_{v \in V} cost(v)$ .

Adding sensors amounts to add terminal constraints (see definition 7). Indeed, as mentioned in section 3, a sensor measuring a variable  $v$  is modeled by the constraint:  $val(t, v) = v$  where  $val(t, v)$  is a data coming from the sensor. Therefore, structurally speaking, a sensor measuring  $v$  is modelled by a terminal constraint  $k$  satisfying  $var(k) = \{v\}$ . The constraint  $k$  will be denoted:  $k_{sensor}(v)$ . By extension, the terminal constraints modelling sensors measuring variables  $V$  are denoted:  $K_{sensor}(V)$ .

The method to solve these complete specifications can be decomposed into two steps: determination of candidate variables for sensor placements using theorem 1, and reduction of the candidate variables in order to find the minimal cost sensor placement that satisfies the complete diagnosability specifications using a branch and bound algorithm. Figure 8 presents the dependency scheme of the method.

The  $findCandidates()$  (algorithm 8) method is based on theorem 1. It takes into account the specifications to determine a set of variables to be measured. If these variables are measured, the complete specifications will be satisfied. This algorithm depends on the  $findLinkingVariables()$  method, which is given by algorithm 9. This algorithm uses the results issuing from algorithm 5 to find a subset of variables linking a subset of constraints  $K_{\Delta}$ , considering only the variables  $V_{\Delta}$ . In this algorithm, the cost of variables is considered. This algorithm depends on

**Algorithm 8**  $findCandidates(K_{\Sigma}, K_{nondet}^{spec}, \mathbb{K}_{nondis}^{spec}, K_{diag}^{spec})$ :  
A set of variables to be measured to satisfied complete specifications,  $\emptyset$  if no solution

**Require:** Specifications are consistent in  $K_{\Sigma}$

```

 $\mathbb{B} \leftarrow findIsolatedBlocks(K_{\Sigma}, K_{nondet}, var(K_{\Sigma}))$ 
if  $\mathbb{B}.cons = K_{nondet}^{spec}$  then
   $V_{nomes} \leftarrow findStumpVariables(K_{\Sigma}, K_{nondet}, var(K_{\Sigma}))$ 
   $K_{\Delta} \leftarrow K_{\Sigma} \setminus K_{nondet}^{spec}$ 
   $V_{\Delta} \leftarrow var(K_{\Sigma}) \setminus V_{nomes}$ 
  for all  $K \in \mathbb{K}_{nondis}^{spec}$  do
     $V \leftarrow findLinkingVariables(K_{\Delta}, K, V_{\Delta})$ 
    if  $V \neq \emptyset$  then
       $V_{nomes} \leftarrow V_{nomes} \cup V$ 
    else
      return  $\emptyset$ 
    end if
  end for
  return  $var(K_{\Sigma}) \setminus V_{nomes}$ 
else
  return  $\emptyset$ 
end if

```

**Algorithm 9**  $findLinkingVariables(K_{\Sigma}, K_{\Delta}, V_{\Delta}, cost(V_{\Delta}))$ : One set of variables linking  $K_{\Delta}$  in  $K_{\Sigma}$ , in the subset variable  $V_{\Delta}$

```

 $V_{linkedStump} \leftarrow findStumpLinkingVariables(K_{\Sigma}, K_{\Delta}, V_{\Delta})$ 
 $V_{sorted} \leftarrow sortVariables(V_{linkedStump}, cost(V_{linkedStump}))$ 
for  $V \in$  combinations of  $card(K_{\Delta}) - 1$  variables from sorted list  $V_{sorted}$  do
  if  $isInterconnected(K_{\Delta}, V)$  then
    return  $V$ 
  end if
end for
return  $\emptyset$ 

```

the `sortVariables()` method, which sorts a list of variables according to measurement costs according to descending order.

A subset of the candidate variables may also lead to the satisfaction of the specifications. A branch and bound algorithm is used to select the most interesting candidate variables to be measured in order to find an optimal sensor placement. Before defining the optimisation algorithm, it is necessary to be able to check if the complete specifications are satisfied for a given subset of candidate variables. Method `isCSFeasible()` (algorithm 10) achieves this.

---

**Algorithm 10** `isFeasible( $K_\Sigma, V_{measured}, K_{nondet}^{spec}, \mathbb{K}_{nondis}^{spec}, K_{diag}^{spec}$ )`: True is the sensor placement satisfies the specifications

---

**Require:** Specifications are consistent in  $K_\Sigma$

```

 $K_{global} \leftarrow K_\Sigma \cup sensor(V_{measured})$ 
 $(\mathbb{B}_{nondet}, \mathbb{B}_{nondis}, B_{diag}) = findBlocks(K_\Sigma, var(K_\Sigma) \setminus V_{measured})$ 
if  $B_{diag}.cons \neq K_{diag}^{spec}$  then
  return false
else if  $B_{nondet}.cons \neq K_{nondet}^{spec}$  then
  return false
else
  for all  $K^{spec} \in \mathbb{K}_{nondis}^{spec}$  do
     $found \leftarrow false$ 
    for all  $B \in \mathbb{B}_{nondis}$  do
      if  $B.cons = K^{spec}$  then
         $found \leftarrow true$ 
      end if
    end for
    if  $found = false$  then
      return false
    end if
  end for
end if
return true

```

---

The optimality criterion for a feasible sensor placement defined by  $V_{measured}$  is given by:  $cost(V_{measured})$ . The branch and bound search algorithm is implemented in the `placeSensor()` method (algorithm 11) using a simple First In First Out buffer of nodes of variables.

## 8. Application

In this section, the special case of dynamical system modeled by recurrent or differential equations is discussed. Then, an example is presented.

**8.1. Dynamical systems.** Sensor placement method relies on structural modeling. Therefore it should be suitable for most systems. Let's examine the special case of

---

**Algorithm 11** `placeSensor( $K_\Sigma, K_{nondet}^{spec}, \mathbb{K}_{nondis}^{spec}, K_{diag}^{spec}$ )`: a set of variables to be measured

---

**Require:** Specifications are consistent in  $K_\Sigma$

**Require:**  $cost()$  is defined for each variable in  $V_\Sigma$

```

 $criteria \leftarrow cost(var(K_\Sigma))$ 
 $V_{candidate} \leftarrow findCandidates(K_\Sigma, K_{nondet}^{spec}, \mathbb{K}_{nondis}^{spec}, K_{diag}^{spec})$ 
 $V_{measured} \leftarrow V_{candidate}$ 
 $buffer \leftarrow \emptyset$ 
 $buffer.push(Vnode(\emptyset, V_{candidate}))$ 
while  $buffer$  is not empty do
   $Vnode \leftarrow buffer.pop()$ 
   $V_{remaining} \leftarrow Vnode^+$ 
  for all  $v \in Vnode^+$  do
     $V_{selected} \leftarrow Vnode^- \cup \{v\}$ 
    if  $cost(V_{selected}) < criteria$  then
      if isFeasible( $K_\Sigma, V_{selected}, K_{nondet}^{spec}, \mathbb{K}_{nondis}^{spec}, K_{diag}^{spec}$ ) then
         $criteria \leftarrow cost(V_{selected})$ 
         $V_{measured} \leftarrow V_{selected}$ 
      else
         $V_{remaining} \leftarrow V_{remaining} \setminus \{v\}$ 
         $buffer.push(Vnode(V_{selected}, V_{remaining}))$ 
      end if
    end if
  end while
return  $V_{measured}$ 

```

---

dynamical systems. Generally speaking, a model is said to be dynamic if either:

- a variable appears several times in a system but at different timestamps
- or a variable and some of its derivatives or summations (whatever the order is) appear in the system.

The first case mainly concerns time-delays and discrete time recurrent systems. According to section 3, each variable stands for a tube in a phenomenological space. Therefore a time delay, modelled by  $y(t + \Delta) = x(t)$ , is a constraint that establishes a link between two tubes:  $\{dom(y(t + \Delta)); \forall t\}$  and  $\{dom(x(t)); \forall t\}$ . Therefore, even if the two variables models the same phenomenon, in the structural model, they cannot be merged. Consider now the following discrete-time recurrent model:

$$\begin{aligned} x((k+1)T_e) &= Ax(kT_e) + Bu_k(kT_e) \\ y(kT_e) &= Cx(kT_e) \end{aligned} \quad ; k \in \mathbb{N}$$

where  $T_e$  stands for the sampling period.

The phenomenon modelled by  $x$  appears twice. Therefore, the constraint must be implicitly completed by a time delay between variables  $x((k+1)T_e)$  and  $x(kT_e)$ . Structurally speaking, these constraints are modelled by

the following structures:

$$\text{var}(k_1) = \{x(kT_e), x((k+1)T_e), u(kT_e)\}$$

$$\text{var}(k_2) = \{x(kT_e), x((k+1)T_e)\}$$

$$\text{var}(k_3) = \{x(kT_e), y(kT_e)\}$$

Moreover, if the tube corresponding to  $x((k+1)T_e)$  appears only once in this constraints (which is usually the case in practice), constraints  $k_1$  and  $k_2$  can be merged:

$$\text{var}(k_{12}) = \{x(kT_e), u(kT_e)\}$$

$$\text{var}(k_3) = \{x(kT_e), y(kT_e)\}$$

The second case mainly concerns integrations and differential equations. Consider for instance the following model:  $\frac{dx}{dt} = u$ .  $\frac{dx}{dt}$  corresponds to a tube, which can be connected to  $x$  in adding the implicit constraint:  $x = \int \frac{dx}{dt} dt$ . The initial condition could also be taken into account by considering  $x = \int_0^{t_f} \frac{dx}{dt} dt + x_0$ . In this case, the structures of the constraints become:  $\text{var}(k_1) = \{\frac{dx}{dt}, u\}$  and  $\text{var}(k_2) = \{x, \frac{dx}{dt}, x_0\}$ . In the same way as time-delays, the constraints  $k_1$  and  $k_2$  can be merged to obtain the following structure:  $\text{var}(k_{12}) = \{u, x\}$  or, if the initial condition is considered,  $\text{var}(k_{12}) = \{u, x, x_0\}$ . This result remains true for summations and derivatives of any order.

Consequently, these kinds of dynamical systems can be handle just like other systems.

**8.2. Example.** The method presented in this paper has been applied to a sensor placement for an electronic circuit (figure 9). It is modeled by the following constraints:

$$\begin{aligned} k_1 : v_{1c} &= v_2 & k_8 : v_3 - v_{4a} &= R_2 i_2 \\ k_2 : i_1 &= i_2 + i_3 & k_9 : v_1 - v_{4b} &= R_3 i_3 \\ k_3 : v_1 &= v_{1a} & k_{10} : v_2 &= R_4 i_4 \\ k_4 : v_1 &= v_{1b} & k_{11} : v_4 &= v_{4a} \\ k_5 : v_1 &= v_{1c} & k_{12} : v_4 &= v_{4b} \\ k_6 : v_0 - v_1 &= R_1 i_1 & k_{13} : v_0 &= \text{val}(v_0) \\ k_7 : C(v_{1a} - v_3) &= \int_0^t i_2 dt \end{aligned} \quad (5)$$

with  $K_\Sigma = \{k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}, k_{11}, k_{12}, k_{13}\}$ .

The corresponding structural matrix is given by table 1.

Suppose that the costs of the measurements are:

$$\begin{aligned} \text{cost}(v_0) &= \text{cost}(v_1) = \text{cost}(v_2) = \text{cost}(v_3) = \text{cost}(v_4) \dots \\ \dots &= \text{cost}(v_{1a}) = \text{cost}(v_{1b}) = \text{cost}(v_{1c}) = \text{cost}(v_{4a}) \dots \\ \dots &= \text{cost}(v_{4b}) = 1 \end{aligned}$$

and

$$\text{cost}(i_1) = \text{cost}(i_2) = \text{cost}(i_3) = \text{cost}(i_4) = 2$$

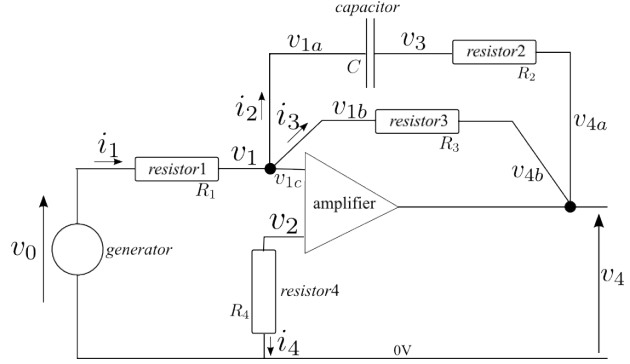


Fig. 9. Scheme of the electronic circuit

Let's consider the following complete specifications:

$$\mathbb{K}_{nondis} = \{\{k_2, k_6\}, \{k_7, k_8\}\}$$

$$K_{nondet} = \{k_1, k_4, k_{10}\}$$

$$K_{diag} = \{k_3, k_5, k_9, k_{11}, k_{12}\}$$

In order to check if the specifications  $K_{nondet}$  are satisfiable, algorithm 2 is used with  $K_\Delta = \{k_1, k_4, k_{10}\}$ ,  $K_\Sigma$  and  $V_\Delta = \text{var}(K_\Sigma)$ . Algorithm 2 computes the following sets of isolated constraints:  $\{\{k_{10}, k_1\}, \{k_4\}\}$ . Specifications  $K_{nondet}$  are consequently satisfiable. Algorithm 2 also provides the isolated variables:  $V_{isolated} = \{i_4, v_{1b}, v_2\}$ .

In order to check if the specifications  $\mathbb{K}_{nondis}$  are satisfiable, algorithm 9 is used with the two subset  $K_{\Delta_1} = \{k_2, k_6\}$  and  $K_{\Delta_2} = \{k_7, k_8\}$ , considering  $V_\Delta = \text{var}(K_\Sigma \setminus V_{isolated})$ . Algorithm 9 computes the two following linking variable subsets:  $V_1 = \{i_1\}$  and  $V_2 = \{v_3\}$ .

In order to find the candidate variables to be measured to satisfy specifications, the algorithm 8 is used. Algorithm 8 computes the following terminal constraints containing the variables from  $\{v_0, v_1, v_4, i_2, i_3, v_{1a}, v_{1c}, v_{4a}, v_{4b}\}$ .

In order to find the cheapest sensor placement that satisfies specifications, algorithm 11 is used. It yields:  $V_{minimal} = \{v_0, v_4, i_2, i_3, v_{1a}, v_{1c}, v_{4a}, v_{4b}\}$  with a cost of 10.

In order to validate the result, the method proposed in (Ploix *et al.*, 2005) has been used to design all the ARR. It has led to the fault signature given by table 2.

According to these results, the constraint set that cannot be discriminated are:  $\{k_2, k_6\}$  and  $\{k_7, k_8\}$ . The constraint set that cannot be detected is:  $\{k_1, k_4, k_{10}\}$  and the diagnosable constraints are:  $\{k_3, k_5, k_9, k_{11}, k_{12}\}$ . Applying the function  $\Phi : K_\Sigma \rightarrow C_\Sigma$ , it is obvious that the components that cannot be discriminated are:  $\{c_2, c_6\}$  and  $\{c_7, c_8\}$ , the components that cannot be detected are:  $\{c_1, c_4, c_{10}\}$  and the diagnosable components are:  $\{c_3, c_5, c_9, c_{11}, c_{12}\}$ .



Suppose now that the specifications are given by:

$$\begin{aligned}\mathbb{K}_{nondis} &= \{\{k_2, k_3\}, \{k_7, k_8\}\} \\ K_{nondet} &= \{k_1, k_{10}\} \\ K_{diag} &= \{k_6, k_4, k_5, k_9, k_{11}, k_{12}\}\end{aligned}$$

In order to check if the specifications  $K_{nondet}$  are satisfiable, algorithm 2 is used with  $K_{\Delta} = \{k_1, k_{10}\}$ ,  $K_{\Sigma}$  and  $V_{\Delta} = var(K_{\Sigma})$ . Algorithm 2 computes the following sets of isolated constraints:  $\{\{k_{10}, k_1\}\}$ . Specifications  $K_{nondet}$  are consequently satisfiable. Algorithm 2 also provides the isolating variables:  $V_{isolated} = \{i_4, v_2\}$ .

In order to check if the specifications  $\mathbb{K}_{nondis}$  are satisfiable, algorithm 9 is used with the two subsets  $K_{\Delta_1} = \{k_2, k_3\}$  and  $K_{\Delta_2} = \{k_7, k_8\}$ , considering  $V_{\Delta} = var(K_{\Sigma} \setminus V_{isolated})$ . Because algorithm 9 computes the linking variable subset:  $V_1 = \{\emptyset\}$  for the constraint subset  $K_{\Delta_1} = \{k_2, k_3\}$ , there are no solution that satisfies these specifications.

The results presented in this paper demonstrate that it is possible to design optimal sensor placements satisfying diagnosability criteria without designing ARR a priori.

## 9. Conclusion

A new approach of sensor placement has been proposed that makes it possible to satisfy diagnosability specifications. It is thus possible to specify the performances that a diagnostic system has to meet and then to compute where the sensors should be placed.

Provided lemmas, theorem and algorithms are general and can be reused to develop other methods for sensor placement that deals with various kinds of specifications, for instance, a set of components that has to be at least detectable and another one that has to be diagnosable. Provided tools apply to any system including dynamical systems depicted by recurrent or differential equations because it is based on a structural approach: only the variables appearing in constraints are considered. However, the generality of structural approach is paid by possible over-estimation depending on the nature of constraints: it is well-known that it relies on the conditioning of constraints. But solutions taking into account the nature of constraints can only be specific.

An algorithm for sensor placement managing complete specifications has been presented. It deals with elements that have to be diagnosable, discriminable and non detectable. Thanks to the proposed algorithm, cost optimal sensor placements satisfying complete diagnosability specifications is possible without designing ARR a priori. It is a very important feature since it is no longer necessary to design all the possible ARR assuming all the variables are measured.

This approach manages only specifications dealing with models of the normal behavior. It does not take

into account specific fault models such as a leak in a pipe. Therefore, if such models are considered, the sensor placement algorithm will lead to an over-estimation of the required sensors. Taking into account specific fault models may lead to a reduction of the required sensors. Nevertheless, fault models cannot be easily taken into account in sensor placement methods. It is still an open problem.

## References

- Apt, K. (2003). *Principles of Constraint Programming*, Cambridge University Press.
- Blanke, M., Kinnaert, M., Lunze, J. and Staroswiecki, M. (2006). *Diagnosis and Fault-tolerant Control*, Springer-Verlag.
- Bollobás, B. (1998). *Modern graph theory*, Graduate Texts in Mathematics, Springer, New York, U.S.A.
- Cassar, J. and Staroswiecki, M. (1997). A structural approach for the design of failure detection and identification systems, *IFAC, IFIP/IMACS Conference on control of industrial Systems*, Belfort, France, pp. 329–334.
- Chittaro, L. and Ranon, R. (2004). Hierarchical model-based diagnosis based on structural abstraction, *Artificial Intelligence 1-2*: 147–182.
- Commault, C., Dion, J.-M. and Yacoub Agha, S. (2006). Structural analysis for the sensor location problem in fault detection and isolation, *SAFEPROCESS'2006*, Beijing, China.
- Console, L., Picardi, C. and Ribando, M. (2000). Diagnosis and diagnosability analysis using process algebra, *DX'2000*, Morelia, Mexico.
- Cordier, M.-O., Dague, P., Lévy, F., Dumas, M., Montmain, J., Staroswiecki, M. and Travé-Massuyès, L. (2000). A comparative analysis of ai and control theory approaches to model-based diagnosis, *SAFEPROCESS 2000*, Budapest, Hungary, pp. 329–334.
- Dalton, T., Klotzek, P. and Frank, P. (1999). Application of sensitivity theory to fuzzy logic based fdi, *International Journal of Applied Mathematics and Computer Science 9*(3).
- de Kleer, J. and Williams, B. C. (1992). Diagnosis with behavioral modes, *Readings in model-based diagnosis* pp. 124–130.
- Dulmage, A. L. and Mendelsohn, N. S. (1959). A structure theory of bi-partite graphs of finite exterior extension, *Transactions of the Royal Society of Canada 53*(III): 1–13.
- Frisk, E. and Krysander, M. (2007). Sensor placement for maximum fault isolability, *The 18th International Workshop on Principles of Diagnosis (DX-07)*.
- Hamscher, W., Console, L. and De Kleer, J. (1992). *Readings in Model-Based Diagnosis*, Morgan Kaufmann.
- Korbicz, J., Patan, K. and Obuchowicz, A. (1999). Dynamic neural networks for process modelling in fault detection and isolation, *International Journal of Applied Mathematics and Computer Science 9*(3).

- Koscielny, J., Syfert, M. and Bartys, M. (1999). Fuzzy logic fault diagnosis of industrial process actuators, *International Journal of Applied Mathematics and Computer Science* **9**(3).
- Krysander, M., Aslund, J. and Nyberg, M. (2005). An efficient algorithm for finding over-constrained sub-systems for construction of diagnostic tests, *16th International Workshop on Principles of Diagnosis (DX-05)*.
- Lopez-Toribio, C., Patton, R. and Uppal, F. (1999). Artificial intelligence approaches to fault diagnosis for dynamic systems, *International Journal of Applied Mathematics and Computer Science* **9**(3).
- Madron, F. and Veverka, V. (1992). Optimal selection of measuring points in complex plants by linear models, *AIChE* **38**(2): 227–236.
- Maquin, D., Luong, M. and Ragot, J. (1997). Fault detection and isolation and sensor network design, *European Journal of Automation* **31**(2): 393–406.
- Nyberg, M. and Krysander, M. (2003). Combining ai, fdi, and statistical hypothesis-testing in a framework for diagnosis., *IFAC Safeprocess '03*, Washington, U.S.A.
- Ploix, S., Désinde, M. and Touaf, S. (2005). Automatic design of detection tests in complex dynamic systems, *16th IFAC World Congress*, Prague, Czech republic.
- Ploix, S., Touaf, S. and Flaus, J. M. (2003). A logical framework for isolation in fault diagnosis, *SAFEPRO-CESS'2003*, Washington D.C., U.S.A.
- Pothen, A. and Chin-Ju, F. (1990). Computing the block triangular form of a sparse matrix, *ACM Transactions on Mathematical Software* **16**(4): 303–324.
- Pulido, B. and Alonso, C. (2002). Possible conflicts, arrs, and conflicts, *13th International Workshop on Principles of Diagnosis (DX02)*, pp. 122–128.
- Shumsky, A. (2007). Redundancy relations for fault diagnosis in nonlinear uncertain systems, *International Journal of Applied Mathematics and Computer Science* **17**(4).
- Struss, P. (1992). What's in sd? towards a theory of modeling for diagnosis, *Readings in model-based diagnosis*, Morgan Kaufman.
- Struss, P., Rehfus, B., Brignolo, R., Cascio, F., Console, L., Dague, P., Dubois, P., Dressler, O. and Millet, D. (2002). Model-based tools for the integration of design and diagnosis into a common process- a project report, *DX'02*, Semmering, Austria.
- Travé-Massuyès, L., Escobet, T. and Milne, R. (2001). Model-based diagnosability and sensor placement application to a frame 6 gas turbine subsystem, *12th Int. Workshop on principles of diagnosis*, Sansicario, Via Lattea, Italie, pp. 205–212.
- Witczak, M. (2006). Advances in model-based fault diagnosis with evolutionary algorithms and neural networks, *International Journal of Applied Mathematics and Computer Science* **16**(1).