Direct Method for Structural Observability Analysis

Ignacio Ponzoni,[†,‡] Mabel C. Sánchez,[‡] and Nélida B. Brignole*[,†,‡]

Grupo de Investigación y Desarrollo en Computación Científica (GIDeCC), Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur (UNS), Avenida Alem 1253, 8000 Bahía Blanca, Argentina, and Planta Piloto de Ingeniería Química (PLAPIQUI), UNS-CONICET, Complejo CRIBABB, Km 7, Camino La Carrindanga, CC 717, 8000 Bahía Blanca, Argentina

A noncombinatorial method for structural observability analysis is presented in this paper. The technique rearranges the process occurrence matrix to a specific block lower-triangular pattern by means of bigraphs and digraphs in two consecutive stages. The algorithmic core is constituted of a new node classification that leads to suitable maximum-matching decompositions even for structurally singular matrices. A three-step strategy for the identification and analysis of forbidden subsets was also designed to take into account the additional numeric constraints that guarantee further solvability of the final pattern. In contrast with other structural techniques, the proposed method treats complex nonlinear models in a remarkably efficient way. Its performance was compared with existing structural observability techniques for three industrial problems. The final results revealed that the direct method is extremely robust and efficient in computing times, becoming more efficacious as problems grow in size and complexity.

1. Introduction

The observability analysis consists in classifying the unmeasured process variables in order to determine the observable ones, i.e., those that can be estimated through model equations using measured values. The analysis is frequently carried out on the steady-state mathematical model chosen to represent plant behavior, which is typically a nonlinear system of algebraic equations expressed in terms of measured and unmeasured variables. The categorization procedure is an essential stage in process plant instrumentation design. A correct rigorous analysis avoids the inclusion of an extra amount of measurements in order to guarantee the required knowledge of the plant state.

There are two main approaches to carry out the classification of unmeasured variables. One of them is topology-oriented because the analysis is based on the identification of cutsets and cycles within the undirected graph that represents process topology. In contrast, the other one is equation-oriented, making use of the system of algebraic equations that constitutes the mathematical model of the plant under study in various ways. In the first category, the most representative papers[1−3] address the topology-oriented approach for linear and bilinear models. In turn, the equation-oriented techniques can be divided in two subcategories: nonstructural and structural methods. The former are numerical procedures[4,5] that make use of different matrices associated with the plant's standard operating point. The latter[6−8] make do with the information provided by the occurrence matrix, which is a binary array that indicates all of the process variables that are present in each model equation. One of the main advantages of the structural approach is its independence from individual operating points, together with its potential capacity to deal with nonlinear plant models rigorously. A thorough critical overview of the existing methodologies for the categorization of unmeasured variables was presented in a paper by Ponzoni et al.,[8] where we proposed a robust structural technique called GS-FLCN that could be applied to nonlinear systems of equations.

GS-FLCN proved to be the most adequate alternative to ensure reliable results for the rigorous analysis of complex models. Nevertheless, its combinatorial nature constitutes a disadvantage for the treatment of large problems, mainly because computing times grow exponentially with problem size. This drawback can be overcome to a certain extent by means of either judicious tree pruning (FLCN with BF[8]) or parallel processing (p-FLCN[9]). Both strategies succeed in shifting the practical limit of manageable problems significantly at the expense of some loss of effectiveness in the case of FLCN with BF or additional hardware requirements for p-FLCN.

In this paper, we present a direct technique for observability analysis that is based on graph theory and matrix permutations. In contrast with GS-FLCN and its variants, this method is noncombinatorial, thus being much more efficient in terms of computing times. The direct method has an enormous potential for its application to plantwide rigorous instrumentation design because it keeps GS-FLCN's robustness while providing much more efficiency with regard to execution times. Therefore, it is a powerful procedure to be incorporated into a package for industrial-scale applications.

As to the organization of the paper, sections 2 and 3 contain the fundamentals of structural observability analysis and graph theory that are required to understand the description of the direct method that follows in section 4. Then, three industrial examples are discussed in section 5, where the performance of the direct method is contrasted with GS-FLCN's performance. Finally, the conclusions are summarized in section 6. An appendix with the main algorithms has also been included to strengthen the understanding of the proposal.

* To whom correspondence should be addressed. E-mail: dybrigno@criba.edu.ar.
[†] Universidad Nacional del Sur.
[‡] UNS-CONICET.

10.1021/ie0300326 CCC: $27.50
Published on Web 12/17/2003

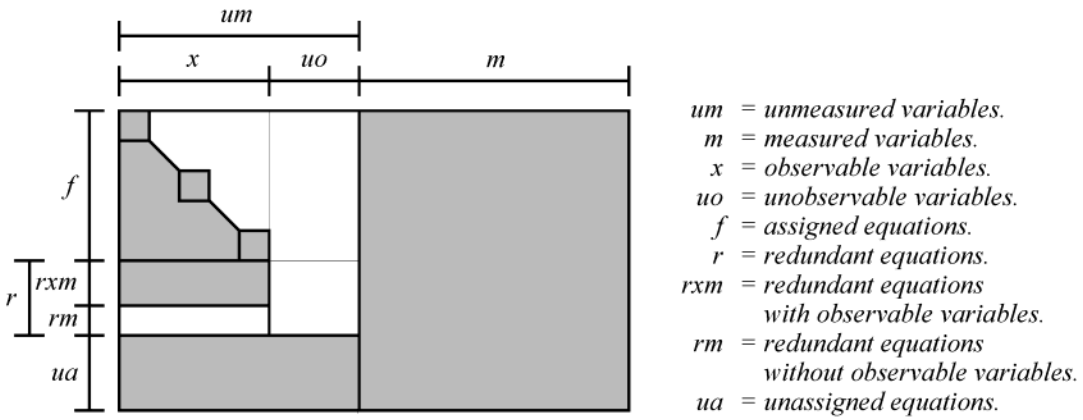


Figure 1. Block arrangement required for the observability classification.

2. Structural Approach for Observability Analysis

There are two main approaches for the development of algorithms for the classification of process variables, namely, strategies based on either process topology or model equations. Our contributions, which fall into the second category, belong to the field of structural observability analysis, which is basically a nonnumeric equation-oriented approach for the classification of unmeasured variables.

The interactions among process variables are described through the mathematical model chosen for the process under study. The associated occurrence matrix **M**, whose rows and columns represent the model equations and process variables, respectively, provides the key information required for the analysis. This matrix is always sparse, with its nonzero entries indicating that the variable associated with the column is present in the equation described in that row. It is interesting to note that the elements in **M** are unaffected by changes in model coefficients, with their typical values being either 0 or 1. The nonnumeric nature of the approach is advantageous because it allows a high degree of independence from the operating points, thus leading to more general results.

Prior to the analysis, the columns of the occurrence matrix are permuted to yield a two-block partitioning: $\mathbf{M} = [M_1, M_2]$, where the columns of M_1 correspond to the unmeasured variables and those of M_2 are associated with the measurements. The observability analysis is performed exclusively on M_1, which is typically rectangular in shape and structurally nonsingular. The core classification procedure for the structural approach is an algorithm that permutes the occurrence matrix to the desired pattern, which evidences the classification (see Figure 1). The shaded zones correspond to the parts of the matrix that contain nonzero entries. The pattern includes a block lower-triangular area that comprises all of the observable variables, i.e., those whose values could be calculated from the measurements. The diagonal blocks in this area are called assignment subsets. The block lower-triangular form (BlTF) reveals a precedence order for the calculations because the assignment subsets can be solved sequentially in blocks. The square subsystems of equations indicated by the assignment subsets should be consistent in order to serve as a means of calculating the assigned process variables. This aspect, which is inherently numerical, is contemplated through an allowability test[8] that discards the subsets that correspond to spurious cases such as parallel streams as well as undesirable combinations whose numerical solution would be extremely cumbersome. The unmeasured variables that remain unobservable are indeterminable for the given set of measurements.

Regarding the application of the observability analysis to process plant instrumentation or revamp, it should be noted that the equations that contain those unknowns constitute a subsystem with infinitely many solutions. Particular solutions can be reached by adding new measurements because the addition of any single measurement might lead to further decoupling. After measurement incorporation, the whole classification procedure should be carried out again to ensure the best BlTF for the modified set of measurements.

In this paper, we present a direct method to carry out the core classification task more efficiently. The quality of the technique was assessed by comparison with GS-FLCN's quality[8] because it was the most robust tool available to date for the rigorous treatment of complex problems. In sharp contrast with the new proposal, GS-FLCN employs a costly combinatorial depth-first search (DFS) procedure to rearrange the occurrence matrix to the BlTF that evidences the classification. Moreover, for large size problems that contain assignment subsets of high order, such as the third example discussed below, GS-FLCN can reach a complete classification only when the branching factor (BF) acceleration technique is used. The BFs reduce the search space by limiting the amount of branches to be explored while seeking assignment subsets of a given order. The time complexity of the FLCN algorithm increases exponentially with problem size.[9] Therefore, the use of BFs together with GS-FLCN proved to be indispensable in practice, despite being detrimental to effectiveness because of the fact that some branches of the tree are never reached. Although the risk of skipping over assignment subsets cannot be avoided completely, it is, in fact, much lower for the expert user because his insight into the physical problem helps him a lot at the decision-making stages of the procedure.

3. Fundamentals

In this section, we define the concepts required to understand the description of the direct method given in section 4. The interested reader will find detailed coverage on graph theory and its applications in many textbooks.[10−14]

3.1. Some Basic Definitions. A *graph* $\boldsymbol{G} = (\mathcal{N}, \mathcal{E})$ comprises a finite set of *nodes* $\mathcal{N}$ and a finite set of

ordered pairs $\mathcal{E}$, known as *edges*. The *order n* of a graph $\boldsymbol{G}$ is its number of nodes. A *walk* between nodes u and v along a graph $\boldsymbol{G}$ is a finite alternating sequence $u = n_0, e_1, n_1, e_2, n_2, e_3, \ldots, e_k, n_k = v$ of nodes and edges in $\boldsymbol{G}$, so that edge e_i joins nodes n_{i-1} and n_i. The nodes and edges in a walk are not necessarily different. A walk that does not contain repeated nodes is called a *path*. A pair of nodes in a graph is a *connected pair* if there is a path between them. A graph $\boldsymbol{G}$ is a *connected graph* if every pair of nodes in $\boldsymbol{G}$ is a connected pair.

A *directed graph* $\boldsymbol{D}$ (also named *digraph*) $\boldsymbol{D} = (\mathcal{N}, \mathcal{E})$ is a graph where each edge $e = (n_0, n_1)$ implies a connection directed from node n_0 to node n_1. In this case, n_0 and n_1 are the *end points* of edge e, and it is said that the nodes are *connected*. When two or more edges connect the same pair of nodes, those edges are called *parallel edges*. A *loop* is an edge whose end points coincide. $\boldsymbol{D}$ is a *simple graph* if it does not contain loops or parallel edges. It is said that $\boldsymbol{D}$ is *strongly connected* if there is a path from u to v and there is also another path from v to u for any pair of distinct nodes $u, v \in \mathcal{N}$. Digraphs that are not strongly connected can always be divided into strongly connected subgraphs known as $\boldsymbol{D}$'s *strong components*.

3.2. Connectivity and Depth-First Searches on Digraphs. An *acyclic graph* (also known as *forest*) is a graph with no cycles. A *tree* is a connected acyclic graph. An acyclic connected subgraph of $\boldsymbol{D}$ is called a *spanning tree* in $\boldsymbol{D}$.

A graph exploration technique that proceeds to successive levels in a tree at the earliest possible opportunity is called a DFS. A series of n DFSs that start from each node in $\boldsymbol{D}$ is called a *transversal* DFS on $\boldsymbol{D}$. If it is found that a digraph $\boldsymbol{D}$ of order n is a connected graph by using DFSs, the set of $n - 1$ used edges in $\boldsymbol{D}$ constitute the edges of a spanning tree called a **DFS** *spanning tree* in digraph $\boldsymbol{D}$. The set of trees that can be generated for digraph $\boldsymbol{D}$ constitutes the DFS *spanning forest* $\boldsymbol{F}$ in $\boldsymbol{D}$. Hence, it is possible to split the set $\mathcal{E}$ into a subset $\mathcal{E}_{\mathcal{F}}$ whose elements are all of the edges used in the spanning forest and a complementary subset $\mathcal{R} = \mathcal{E} - \mathcal{E}_{\mathcal{F}}$ that contains the rest of the edges. The edges in $\mathcal{R}$ are called *backward edges* because they connect nodes with their ancestors.

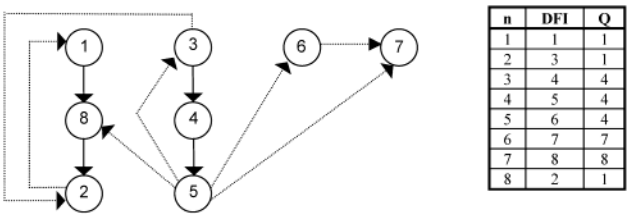
For digraphs, the spanning forest is said to be *outgoing* because the heads of all of the edges in the forest point at the leaves. In this case, the set of edges A is partitioned into four subsets as follows: $\mathcal{E}_{\mathcal{F}}$ (*outgoing edges*) = {edges that belong to $\boldsymbol{F}$}, $\mathcal{R}_1$ (*backward edges*) = {edges that go from a node to one of its ancestors}, $\mathcal{R}_2$ (*forward edges*) = {edges that do not belong to $\boldsymbol{F}$ and go from a node to one of its descendants}, and $\mathcal{C}$ (*cross edges*) = {edges that connect pairs of nodes from different trees in the forest}.

Let us define the label DFI(v) as a nonnegative integer that indicates the order of node v in the sequence of visited nodes during a DFS. DFI(v) = 0 for any node that has not been visited yet.

The following basic theorem holds:

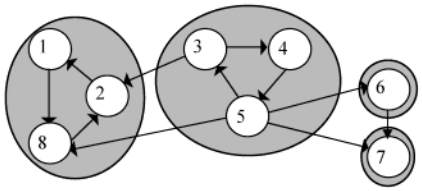
Theorem 1.[10] If $(u, v) \in \mathcal{C}$, then DFI(u) > DFI(v) during a DFS on a digraph.

3.3. Strong-Component Detection. The direct method presented in this paper involves matrix decompositions and rearrangements that are determined from an analysis of the strong components in digraphs associated with certain blocks in M_1. The most efficient algorithm for the detection of strong components[15] is based on a DFS procedure. The method fundamentally depends on the following statement:



n	DFI	Q
1	1	1
2	3	1
3	4	4
4	5	4
5	6	4
6	7	7
7	8	8
8	2	1

a) $\boldsymbol{G}$'s spanning forest; DFI and Q values for each node.



b) The strong components in graph $\boldsymbol{G}$.

Figure 2. Detection of all of the strong components of a digraph.

Theorem 2.[10] If $\boldsymbol{D}_i = (\mathcal{N}_i, \mathcal{E}_i)$ is a strong component of a digraph $\boldsymbol{D}$ and $\boldsymbol{F} = (\mathcal{N}_{\mathcal{F}}, \mathcal{E}_{\mathcal{F}})$ is a DFS outgoing spanning forest, then $\boldsymbol{T}_i = (\mathcal{N}_i, \mathcal{E}_i \cap \mathcal{E}_{\mathcal{F}})$ is a tree.

$\boldsymbol{T}_i$'s root r_i is called the *root of* $\boldsymbol{D}_i$*'s strong component*. From theorem 2, it is possible to find the strong components of a digraph $\boldsymbol{D}$ by obtaining the roots $r_1, r_2, \ldots, r_k$, in a convenient order. During the transversal DFS on $\boldsymbol{D}$, r_i will be visited for the last time before the last visit to r_j if $i < j$. Then, from theorem 1 and also considering that r_j cannot be a descendant of r_i's if DFI(r_i) > DFI(r_j), it can be deduced that $\boldsymbol{D}_i$ is a subgraph induced by all of the nodes in $\boldsymbol{D}$ that are only descended from r_i, without being descendants of $r_1, r_2, \ldots, r_{i-1}$.

Parameter $Q(v)$ will be defined next in order to facilitate the computational identification of the roots of the strong components.

$$Q(v) = \min(\{\mathrm{DFI}(v)\} \cup \{\mathrm{DFI}(v') | (x, v') \text{ belongs either to } \mathcal{R}_1 \text{ or to } \mathcal{C}\}) \quad (1)$$

where x is a descendant of v and the root r of the strong component that contains v' is an ancestor of v.

The following theorem constitutes the basis for an algorithm that detects the roots of the strong components in a digraph.

Theorem 3.[10] v is the root of a strong component in digraph $\boldsymbol{D}$ if and only if $Q(v) = \mathrm{DFI}(v)$.

The detection algorithm recursively evaluates $Q(v)$ in order to embed it into the DFS. This is done by reformulating $Q(v)$ as follows:

$$Q(v) = \min(\{\mathrm{DFI}(v)\} \cup \{Q(v') | v' \text{ is a son of } v\text{'s}\} \cup \{\mathrm{DFI}(v') | (v, v') \text{ is either in } \mathcal{R}_1 \text{ or in } \mathcal{C}, \text{ with the root of the strong component that contains } v' \text{ being an ancestor of } v\text{'s}\}) \quad (2)$$

The complete procedure can be explained through two pseudoalgorithms that are reported in the appendix (algorithms A.1 and A.2). There is a basic algorithm (SSC) that finds whether a given node is the root of a strong component by means of eq 2. This routine is called by the main program (SC) in order to detect all of the strong components in a digraph.

Figure 2 shows how to apply these algorithms to digraph $\boldsymbol{D} = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N} = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and

$\mathcal{E} = \{(1, 8), (2, 1), (3, 2), (3, 4), (4, 5), (5, 3), (5, 6), (5, 7), (5, 8), (6, 7), (8, 2)\}$. The outgoing spanning forest for this example is presented in Figure 2a, together with a table that reports the DFI and Q values assigned to each node during the searches. The full lines represent the edges that belong to $\mathcal{E}_\mathcal{F}$, while the edges in $\mathcal{R}_1$, $\mathcal{R}_2$, or $\mathcal{C}$ are plotted in dotted lines. The final results are shown in Figure 2b, where each shaded circle indicates a strong component in $\boldsymbol{D}$.

3.4. Bipartite Graphs and Maximum Matching. A *bipartite graph* (also named *bigraph*) $\boldsymbol{B} = (\mathcal{N}_1, \mathcal{N}_2, \mathcal{E})$ is a simple graph whose nodes can be partitioned in two disjoint subsets $\mathcal{N}_1$ and $\mathcal{N}_2$, so that each edge in the graph has one end point in each subset.

Given a bigraph $\boldsymbol{B} = (\mathcal{N}_1, \mathcal{N}_2, \mathcal{E})$, a *matching* $\mathcal{P}$ is a subset from $\mathcal{E}$ whose edges have no common end points. The number of edges in $\mathcal{P}$ is called the *cardinality* of the matching $\boldsymbol{C}$. In particular, $\mathcal{P}_i$ is a *maximum matching* for $\boldsymbol{B}$ if there are no other matchings $\mathcal{P}_j \neq \mathcal{P}_i$ in $\boldsymbol{B}$ whose cardinality is higher than $\mathcal{P}$s.

An *alternating walk* along bigraph $\boldsymbol{B}$ related to a matching $\mathcal{P}$ is a walk whose edges alternate in $\mathcal{P}$. In other words, if the first edge in the walk belongs to $\mathcal{P}$, then all of the edges at odd locations in the walk also belong to $\mathcal{P}$, while those at even positions do not. An *alternating path* is an alternating walk without repeated edges, and an *augmenting path* is an alternating path whose terminal edges do not belong to $\mathcal{P}$.

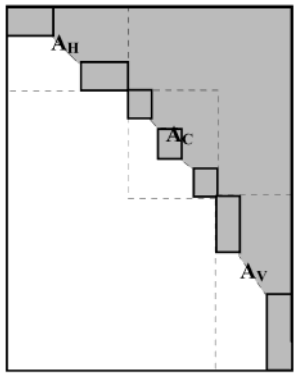
Given a matching $\mathcal{P}$, an augmenting path $\mathcal{A}$ can be employed to obtain a matching $\mathcal{P}'$ whose cardinality is one unit higher than $\mathcal{P}$s. $\mathcal{P}'$ is generated by removing from $\mathcal{P}$ all of the edges that are present in both sets, i.e., the edges in $\mathcal{A} \cap \mathcal{P}$, and adding to $\mathcal{P}$ all of the edges that only appear in $\mathcal{A}$. For a bigraph of interest and starting from a proposed matching $\mathcal{P}$, this procedure can be repeated sequentially until no more augmenting paths can be found. At this point, it is said that a *maximum matching* $\mathcal{P}_\mathrm{m}$ has been reached. It is clear that several maximum matchings can be obtained from a given matching $\mathcal{P}$, depending on the sequence of augmenting paths chosen to build the succession of matchings. Nevertheless, the cardinality of the maximum matchings will always be the same, whatever the generation order. Algorithm A.3 (see the appendix) shows the steps required to obtain a maximum matching for a bigraph $\boldsymbol{B}$.

3.5. Matrix Permutations Using Graphs. The direct method is based on DFSs along bipartite and directed graphs derived from M_1. This matrix is not necessarily square and almost always structurally singular. Therefore, the existing techniques to carry out matrix permutations to block triangular forms (BTFs) proved to be unsuitable in this context, and a new algorithm had to be devised. Some traditional methods were not applicable to rectangular structurally singular matrices, while others led to patterns with a useless block distribution. In particular, Tarjan[15] proposed a very efficient algorithm for the determination of the strong components of a digraph. The method can be applied to the permutation of matrices to BTFs, provided the matrices have a full transversal. A square matrix is said to have a full transversal when none of its diagonal elements is equal to zero. Later, Gustavson[16] and Duff and Reid[17] developed implementations of Tarjan's method for square matrices.

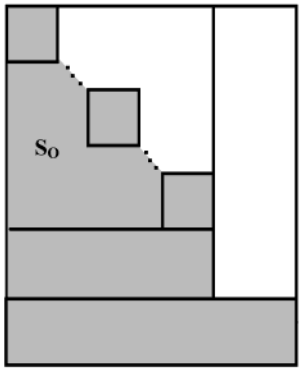
In the case of rectangular structurally singular matrices, the available techniques[18−20] do not yield the desired pattern, leading to block upper triangular shapes. The classic Dulmage−Mendelsohn (DM) decomposition[18,19] is one of the most widespread noncombinatorial techniques to permute a general matrix, i.e., one that is not necessarily square, to a BTF. In brief, the reordering procedure for a given matrix $\mathbf{A}$ comprises two sequential stages. First, a bipartite graph $\boldsymbol{B}$ is associated with $\mathbf{A}$. Then, a maximum matching $\mathcal{P}_\mathrm{m}$ is found for $\boldsymbol{B}$, and the nodes in $\boldsymbol{B}$ are classified as a function of $\mathcal{P}_\mathrm{m}$. Next, $\mathbf{A}$'s rows and columns are rearranged in accordance with the following node classification that was proposed by Dulmage and Mendelsohn: $\mathcal{VR}$ = {nodes in $\mathcal{N}_1$ that can be reached from an unmatched node in $\mathcal{N}_1$ by moving along an alternating path}, $\mathcal{HR}$ = {nodes in $\mathcal{N}_1$ that can be reached from an unmatched node in $\mathcal{N}_2$ by moving along an alternating path}, $\mathcal{SR} = \mathcal{N}_1 \setminus (\mathcal{VR} \cup \mathcal{HR})$, $\mathcal{VC}$ = {nodes in $\mathcal{N}_2$ that can be reached from an unmatched node in $\mathcal{N}_1$ by moving along an alternating path}, $\mathcal{HC}$ = {nodes in $\mathcal{N}_2$ that can be reached from an unmatched node in $\mathcal{N}_2$ by moving along an alternating path}, and $\mathcal{SC} = \mathcal{N}_2 \setminus (\mathcal{VC} \gg \mathcal{HC})$.

As a result, $\mathbf{A}$ is permuted to the BTF shown in Figure 3a, where block $\mathbf{A}_\mathrm{C}$ is always square and has a full transversal.

In 1990, Pothen and Fan[20] introduced a two-stage partitioning technique (PF) for general matrices. In this technique, DM's strategy is applied first in order to yield a coarse-grain decomposition. Then, a fine-grain decomposition is performed. This stage consists in partitioning the three diagonal blocks $\mathbf{A}_\mathrm{H}$, $\mathbf{A}_\mathrm{C}$, and $\mathbf{A}_\mathrm{V}$ into block upper-triangular forms whose diagonal blocks are *irreducible*. A diagonal block is irreducible when it cannot be decomposed into blocks of smaller size without



a) DM's final pattern (dotted lines);



b) BITF required for observability analysis

Figure 3. BTF patterns. Shaded blocks: PF's fine-grain decomposition.

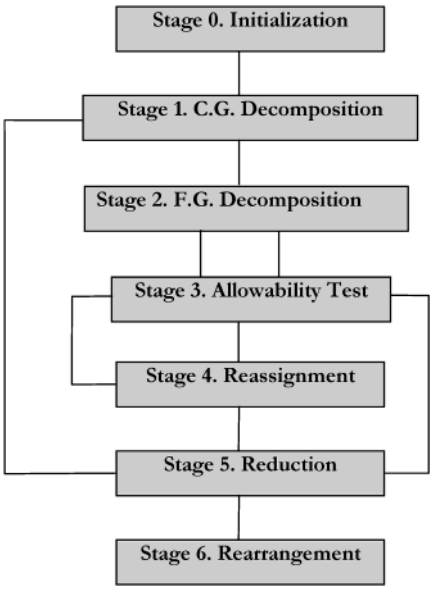


Figure 4. General scheme of the direct method.

destroying the block upper-triangular pattern. Matrices $\mathbf{A}_H$ and $\mathbf{A}_V$ are decomposed by determining all of their connected components by means of a DFS. In turn, block $\mathbf{A}_C$ is partitioned using Tarjan's method for square matrices.[15] The corresponding pattern is shown in Figure 3a.

It is clear from Figure 3 that PF's rearrangement significantly differs from the BlTF required by our structural methodology for observability analysis. On the one hand, PF's pattern is block *upper*-triangular, where only the blocks corresponding to $\mathbf{A}_C$ are square, while the rest ($\mathbf{A}_H$ and $\mathbf{A}_V$) are rectangular. On the other hand, the pattern required for observability analysis is block *lower*-triangular, where all of the blocks on the main diagonal of submatrix $\mathbf{S}_O$ are square. It is evident that it is impossible to transform any of these BTFs into the other one by simple matrix-transpose operations. In this paper, we present a methodology that makes use of the fundamental PF concepts in order to obtain the BlTF required for observability analysis in a direct way.

4. Direct Method for Observability Analysis

This technique, which is based on graph decomposition, is a novel noncombinatorial algorithm for the structural classification of unmeasured variables that is significantly more efficient than FLCN and its variations.[9,21] The main idea is based on the analysis of existing direct methods for matrix reordering that were developed in order to solve systems of linear equations.[17,20] Although both the goal and the scope of those techniques are completely different from ours, the philosophy employed for the structural rearrangement served as a basis for this development.

4.1. General Structure. The algorithm consists of six stages plus an initialization phase, as shown in Figure 4. In the first place, the bigraph $\boldsymbol{B}$ associated with the occurrence submatrix M_1 is built. Then, a *coarse-grain decomposition* of $\boldsymbol{B}$ is carried out. This decomposition involves two consecutive stages: a search for a maximum matching in $\boldsymbol{B}$, followed by an appropriate node classification. Next, a *fine-grain partitioning* is performed in stage 2, which results in the determination of M_1's assignment subsets. All of them are tested for allowability in stage 3. When a block proves to be unallowable, a reassignment is attempted (stage 4); i.e., the process tries to substitute one of the rows in the rejected subset by another row that corresponds to a redundant equation. If the reassignment was successful, the algorithm goes back to step 2. Otherwise, it moves to step 5 (reduction stage). In this phase, all of the nodes associated with the rows and columns that passed the allowability test are removed from $\boldsymbol{B}$. Then, the algorithm goes back to stage 1 in order to process the reduced bigraph. The complete cycle continues until no new assignment subsets are detected. When this is the case, the algorithm moves to stage 6, where the matrix is rearranged and the procedure ends. Figure 4 summarizes the information flow for the direct method, while the complete algorithm is reported in appendix A.4.

4.2. Coarse-Grain Decomposition. The input information required for this decomposition is the bipartite graph $\boldsymbol{B} = (\mathcal{N}_1, \mathcal{N}_2, \mathcal{E})$ associated with the occurrence submatrix M_1, built during the initialization stage. The nodes in $\mathcal{N}_1$ and $\mathcal{N}_2$ correspond to M_1's rows and columns, respectively, while $\mathcal{E}$'s edges represent its nonzero elements. Once $\boldsymbol{B}$ has been generated, the first partitioning level, which we have called *coarse-grain decomposition*, is carried out. In this stage, a maximum matching $\mathcal{P}_m$ is looked for in $\boldsymbol{B}$ by means of the **MM** algorithm (see appendix A.3). Next, the nodes are classified according to $\mathcal{P}_m$.

As pointed out in the Introduction, DM's classification for the nodes in $\boldsymbol{B}$ does not lead to the pattern of interest. Therefore, to modify the categorization to suit our special needs, we devised a different way of categorizing $\boldsymbol{B}$'s nodes. The following proposal of a new node classification was first presented in 1997[22] and later published in Ponzoni et al.[23] The new technique partitions sets $\mathcal{N}_1$ and $\mathcal{N}_2$ into the following disjoint subsets: $\mathcal{VR}$ = {unmatched nodes in $\mathcal{N}_1$}, $\mathcal{SR}1$ = {matched nodes in $\mathcal{N}_1$ reachable from an unmatched node in $\mathcal{N}_1$ along an alternating path}, $\mathcal{HR}$ = {matched nodes in $\mathcal{N}_1$ reachable from an unmatched node in $\mathcal{N}_2$ along an alternating path}, $\mathcal{SR}2 = \mathcal{N}_1 \setminus (\mathcal{VR} \cup \mathcal{SR}1 \cup \mathcal{HR})$, $\mathcal{SC}1$ = {matched nodes in $\mathcal{N}_2$ reachable from an unmatched node in $\mathcal{N}_1$ along an alternating path}, $\mathcal{HC}$ = {nodes in $\mathcal{N}_2$ reachable from an unmatched node in $\mathcal{N}_2$ along an alternating path}, and $\mathcal{SC}2 = \mathcal{N}_2 \setminus (\mathcal{SC}1 \cup \mathcal{HC})$.

The main difference between DM's decomposition and this new proposal lies in the fact that DM's technique places all of the nodes in $\mathcal{N}_1$ that belong to $\mathcal{VR}$ and $\mathcal{SR}1$ in the same set, while the classification for instrumentation design requires one to make a distinction between them. The contrast between both classifications can be appreciated in Figure 5, where Figure 5b shows how the nodes must be rearranged in order to get the desired BlTF. It is important to note that the blocks ($\mathcal{SC}1$, $\mathcal{SR}1$) and ($\mathcal{SC}2$, $\mathcal{SR}2$), apart from being square, have a full transversal because the nodes in $\mathcal{SC}1$, $\mathcal{SR}1$, $\mathcal{SC}2$, and $\mathcal{SR}2$ belong to the maximum matching.

An example of the resulting pattern is presented in Figure 6. It corresponds to the matrix **N** shown in Figure 6a. The corresponding bipartite graph and the node classification are shown in Figure 6b, while Figure 6c contains the matrix that results after applying the coarse-grain decomposition to **N**. It can be noticed that the nodes in $\mathcal{SC}1$ and $\mathcal{SC}2$ are associated with the observable variables, while those in $\mathcal{HC}$ correspond to the indeterminable variables. In turn, the nodes in $\mathcal{SR}1$ and $\mathcal{SR}2$ represent the assigned equations, the ones in

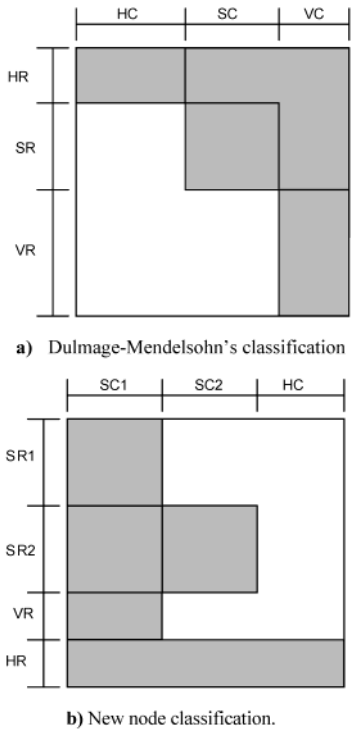


Figure 5. Classification of nodes and the coarse-grain decomposition.

$\mathcal{VR}$ indicate the redundant equations, and those in $\mathcal{HR}$ correspond to equations that contain unobservable variables.

4.3. Fine-Grain Decomposition. The coarse-grain decomposition yields two structurally nonsingular square blocks determined by the sets ($\mathcal{SR}1$, $\mathcal{SC}1$) and ($\mathcal{SR}2$, $\mathcal{SC}2$). The *fine-grain decomposition* stage partitions these blocks into irreducible square subsystems by means of the **SC** algorithm for the detection of the strong components of a digraph (see appendix A.2). First, the digraph $\boldsymbol{D}(\mathbf{N}_{S1})$ is decomposed, where $\mathbf{N}_{S1} = (\mathcal{SR}1, \mathcal{SC}1)$, and then the **SC** algorithm is applied to $\boldsymbol{D}(\mathbf{N}_{S2})$, where $\mathbf{N}_{S2} = (\mathcal{SR}2, \mathcal{SC}2)$. Each strong component that is detected in these digraphs corresponds to an assignment subset in the BlTF.

Figure 7 shows the fine-grain decomposition of block $\mathbf{N}_{S2} = (\mathcal{SR}2, \mathcal{SC}2)$ for the example given in Figure 6. In the first place, the digraph $\boldsymbol{D}(\mathbf{N}_{S2})$ is built (see Figure 7a,b). Each node in $\boldsymbol{D}(\mathbf{N}_{S2})$ is associated with two nodes from the bigraph, which come from $\mathcal{SR}2$ and $\mathcal{SC}2$, respectively. Then, the digraph is partitioned into its strong components (Figure 7c), and finally $\mathbf{N}_{S2}$ is rearranged according to the strong components found through **SC**.

4.4. Allowability Test, Block Reassignment, and Bigraph Reduction. The next algorithmic step consists in verifying whether the assignment subsets found at the fine-grain decomposition stage are acceptable. The allowability test checks all the blocks to make sure that they do not contain any forbidden subsets. The unallowable subsets are specified in the set of constraints **R**, which stores information about subsystems in **N** that have been declared unacceptable for any reason.

First of all, the blocks $\mathbf{N}_{S1i}$, for $i = 1, ..., p$, where p is the number of assignment subsets in $\mathbf{N}_{S1}$, are analyzed. All of these blocks belong to $\mathbf{N}_{S1} = (\mathcal{SR}1, \mathcal{SC}1)$. If a certain block $\mathbf{N}_{S1j}$, with $1 \le j \le p$, contains a forbidden subset **T** listed in **R**, the algorithm moves to the reassignment stage. In this phase, the method looks for a row node r in $\mathcal{VR}$ so that there is an edge (r, c), where c is a column node from $\mathbf{N}_{S1j}$. If such node exists, row node k in $\mathbf{N}_{S1j}$, which formed a pair with c, is replaced by node r. In terms of systems of equations, the reassignment exchanges one of the equations in the subsystem that failed the allowability test for a redundant equation. In Figure 6b, it can be noticed that the reassignment implies a permutation between rows from $\mathcal{SR}1$ and $\mathcal{VR}$.

After permutation, the digraph that corresponds to $\mathbf{N}_{S1}$ is rebuilt and the fine-grain decomposition of this block is carried out again. The objective of the reassignment stage, specially proposed for this algorithm, is to break the structure of block $\mathbf{N}_{S1j}$, which contained the forbidden subset **T**. By exchange of a row node $\mathbf{N}_{S1j}$ for a node in $\mathcal{VR}$, the rejected block $\mathbf{N}_{S1j}$ will not be formed again during the next fine-grain decomposition of $\mathbf{N}_{S1}$.

When it is impossible to reassign block $\mathbf{N}_{S1j}$, the method tries to reduce the bigraph. In this phase, blocks $\mathbf{N}_{S1i}$, for $i < j$, are stored in the structure that keeps the valid assignment subsets. Next, the row and column nodes in those blocks are removed from bigraph $\boldsymbol{B}(\mathbf{N})$. At this point, a row node e from $\mathbf{N}_{S1j}$, which will be called *special row*, is chosen and temporarily removed from $\boldsymbol{B}(\mathbf{N})$. Then, the control is transferred back to the first stage of the method again.

The choice and temporary removal of a special row aims at finding a maximum matching $\mathcal{P}_m'$ that differs from the one obtained from the first application of the coarse-grain decomposition. If no row nodes in $\boldsymbol{B}(\mathbf{N})$ were temporarily removed, the next execution of algorithmic stage 1 would build a maximum matching $\mathcal{P}_m'$ that would be equivalent to $\mathcal{P}_m$, i.e., to the one that had originally been found. The only difference would lie in the fact that $\mathcal{P}_m'$ would not contain those nodes in the blocks $\mathbf{N}_{S1i}$ that had been removed from $\boldsymbol{G}(\mathbf{N})$, with the rest of $\mathcal{P}_m$ remaining equal to $\mathcal{P}_m'$.

After generation of the new maximum matching $\mathcal{P}_m'$, special row e is introduced into the bigraph again and classified according to the following rules:

1. e will belong to $\mathcal{VR}$ if all of its edges lead to columns in $\mathcal{SC}1$ or $\mathcal{SC}2$.

2. e will belong to $\mathcal{SR}1$ if e is connected to only one column in $\mathcal{HC}$ and it is not connected to any columns in $\mathcal{SC}2$ (in this case, this column, together with e, forms a block whose order is 1).

3. e will belong to $\mathcal{SR}2$ if e is connected to only one column in $\mathcal{HC}$ and at least one column in $\mathcal{SC}2$ (in this case, this column, together with e, forms a block whose order is 1).

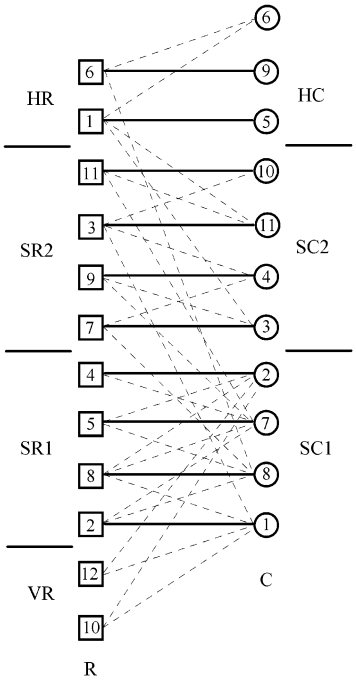
4. e will belong to $\mathcal{HR}$ if e is connected to two or more columns in $\mathcal{HC}$.

Theorem 4. If node e belongs to the rejected block $\mathbf{N}_{S1j}$, this block will not be formed again during the next execution of the fine-grain decomposition procedure.

Proof. This theorem can be proved by the absurd. First of all, it must be taken into account that the constraints always correspond to subsystems whose order is at least 2. Then, if $\mathbf{N}_{S1j}$ has been rejected during the allowability test, its dimension must be greater than 1 because $\mathbf{N}_{S1j}$ contains a constraint. In contrast, the classification rules stated above establish that e can only belong to a block of order 1. As a consequence, $\mathbf{N}_{S1j}$ will not be present in the new fine-grain decomposition.

$$\mathbf{N} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

a) Occurrence Matrix

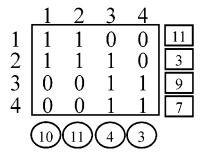


b) Maximum Matching for $\boldsymbol{G}(\mathbf{N})$.

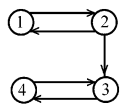
	2	7	8	1	10	11	4	3	6	9	5
4	**1**	1	0	0	0	0	0	0	0	0	0
5	1	**1**	1	0	0	0	0	0	0	0	0
8	1	1	**1**	1	0	0	0	0	0	0	0
2	0	1	1	**1**	0	0	0	0	0	0	0
11	0	1	0	0	**1**	1	0	0	0	0	0
3	0	0	0	1	1	**1**	1	0	0	0	0
9	0	1	0	0	0	0	**1**	1	0	0	0
7	0	0	1	0	0	0	1	**1**	0	0	0
12	1	0	0	1	0	0	0	0	0	0	0
10	1	0	0	1	0	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0	1	**1**	0
1	0	0	0	0	0	1	0	1	1	0	**1**

c) Rearranged Matrix after the Coarse-Grain Decomposition

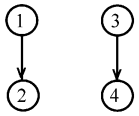
Figure 6. Coarse-grain decomposition.



a) Block $\mathbf{N}_2 = (\mathcal{SC}2, \mathcal{SR}2)$



b) Digraph $\boldsymbol{G}(\mathbf{N}_2)$.



c) Strong Components in $\boldsymbol{G}(\mathbf{N}_2)$.

	3	4	1	2
3	1	1	0	0
4	1	1	0	0
1	0	0	1	1
2	1	0	1	1

d) Block $\mathbf{N}_2$ after Rearrangement.

Figure 7. Fine-grain decomposition of $\boldsymbol{N}_2 = (\mathcal{SC}2, \mathcal{SR}2)$.

In this way, the proposed classification of nodes leads to a new decomposition of $\boldsymbol{B}(\mathbf{N})$, where no nodes are lost because the special row e, which is at first temporarily removed, is reestablished later as soon as the new maximum matching has been found. It is interesting to remark that it is impossible to "lose" an observable variable during the procedure, in principle, because the removal is always temporary. Moreover, the special equation that has been temporarily removed can never contain only one observable variable because if that were the case, that variable and the equation it appears in should have already been detected and classified as a 1×1 diagonal block earlier. In short, this strategy successfully avoids undesirable assignment subsets because the blocks that fail the test are left behind.

After having checked all of the blocks in $\mathbf{N}_{S1}$ successfully, the allowability test is carried out on block $\mathbf{N}_{S2} = (\mathcal{SR}2, \mathcal{SC}2)$. In contrast with $\mathbf{N}_{S1}$, the rejected blocks in $\mathbf{N}_{S2}$ cannot be reassigned. It is clear from the BlTF pattern that the redundant rows only have nonzero elements in the columns that correspond to $\mathcal{SR}1$. Then, the reassignment does not make sense for $\mathbf{N}_{S2}$. There-

fore, whenever a block $\mathbf{N}_{S2i}$ in $\mathbf{N}_{S2}$ is unallowable, the algorithm moves directly toward the bigraph reduction stage.

It sometimes occurs that the removal of only one special row is not enough to find new acceptable blocks. In those cases, the method removes several special rows until some significant progress is achieved in the decomposition; i.e., different row nodes are temporarily removed from the bigraph until new blocks are decoupled. The policy adopted for the classification of the temporarily removed nodes is always the same. Once the new maximum matching has been found, each special row is reincorporated and categorized following the rules stated above.

The algorithm ends either when all of the blocks in $\mathbf{N}_{S2}$ have been accepted by the allowability test or else when the sets $\mathcal{SR}1$ and $\mathcal{SR}2$ returned by the coarse-grain decomposition are empty.

By way of illustration, we shall consider the example in Figure 6 again. The fine-grain decomposition of block $\mathbf{N}_{S1} = (\mathcal{SR}1, \mathcal{SC}1)$ yields only one strong component, which is obviously made up of all of the rows and columns in $\mathbf{N}_{S1}$. Let us assume that there is a constraint **T** in **R** composed of the row nodes {2, 4, 5, 8} and the column nodes {2, 7, 8, 1} because the associated equations and variables correspond to a singular subsystem from the mathematical model of the plant. Then, the block is rejected by the allowability test, and the algorithm proceeds with the reassignment stage. At this point, we shall assume that the row nodes 2 (from $\mathcal{SR}1$) and 10 (from $\mathcal{VR}$) are chosen and exchanged. After that, block $\mathbf{N}_{S1}$ is composed of the row nodes {4, 5, 8, 10}, as shown in Figure 8a, while node 2 corresponds to a redundant equation.

Let us suppose that the fine-grain decomposition is executed again, and as a result, a single strong component that comprises the complete block $\mathbf{N}_{S1}$ is found again. Next, we will assume that the new block, composed of row nodes {4, 5, 8, 10} and column nodes {2, 7, 8, 1}, is checked and rejected as a result of a constraint made up of row nodes {4, 5, 10} and column nodes {2, 7, 8}. This triggers a new reassignment, where, for example, row nodes 4 and 12 are exchanged, yielding a block $\mathbf{N}_{S1}$ with row nodes {5, 8, 10, 12} and column nodes {2, 7, 8, 1}. Next, the fine-grain decomposition is repeated, resulting in the following two strong components: $\mathbf{N}_{S11} = \{\{10, 12\}, \{1, 2\}\}$ and $\mathbf{N}_{S12} = \{\{5, 8\}, \{7, 8\}\}$. Both of them are acceptable, so they pass the allowability test, which ends in this way for $\mathbf{N}_{S1}$. The resulting matrix is shown in Figure 8b.

4.5. Efficiency and Correctness. The order of the proposed algorithm with regard to execution times can be estimated from the order of each decomposition procedure. The coarse decomposition, in particular, employs Hopcroft−Karp's algorithm,[24] whose execution time is $O(n^{3/2}\tau)$ in the worst cases, where n and τ are the number of vertexes and nonzero elements in the matrix, respectively. Because $\tau \leq n^2$, it can be stated that Hopcroft−Karp's algorithm has execution times of $O(n^{5/2})$. As to the fine decomposition, Tarjan's method is of order $(O(n) + O(\tau))$. Taking into account that both methods are applied sequentially, it is immediate that the proposed algorithm has $O(n^{3/2}\tau)$. Finally, it is important to remark that the correctness of our procedure can be assessed from the correctness of Hopcroft−Karp's and Tarjan's algorithms. The methodology is both robust and trustworthy, preserving the high quality of the partitions yielded by those two individual techniques.

	②	⑦	⑧	①	⑩	⑪	④	③	⑥	⑨	⑤
4	**1**	1	0	0	0	0	0	0	0	0	0
5	1	**1**	1	0	0	0	0	0	0	0	0
8	1	1	**1**	1	0	0	0	0	0	0	0
10	1	0	0	**1**	0	0	0	0	0	0	0
11	0	1	0	0	**1**	1	0	0	0	0	0
3	0	0	0	1	1	**1**	1	0	0	0	0
9	0	1	0	0	0	0	**1**	1	0	0	0
7	0	0	1	0	0	0	1	**1**	0	0	0
12	1	0	0	1	0	0	0	0	0	0	0
2	0	1	1	1	0	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0	1	**1**	0
1	0	0	0	0	0	1	0	1	1	0	**1**

a) Partition resulting from the first reassignment.

	①	②	⑦	⑧	⑩	⑪	④	③	⑥	⑨	⑤
10	**1**	1	0	0	0	0	0	0	0	0	0
12	1	**1**	0	0	0	0	0	0	0	0	0
5	0	1	**1**	1	0	0	0	0	0	0	0
8	1	1	1	**1**	0	0	0	0	0	0	0
11	0	0	1	0	**1**	1	0	0	0	0	0
3	1	0	0	0	1	**1**	1	0	0	0	0
9	0	0	1	0	0	0	**1**	1	0	0	0
7	0	0	0	1	0	0	1	**1**	0	0	0
4	0	1	1	0	0	0	0	0	0	0	0
2	1	0	1	1	0	0	0	0	0	0	0
6	0	0	0	1	0	0	0	0	1	**1**	0
1	0	0	0	0	0	1	0	1	1	0	**1**

b) Partitioned matrix after the second reassignment.

Figure 8. Reassignments for block $\mathbf{N}_1$.

5. Performance Analysis

The direct method for observability analysis was employed to classify the unmeasured variables for several industrial problems. The final results revealed that the technique is extremely robust and efficient in computing times. In this section, we present three problems where the performance of the proposed method is assessed by comparison with the best existing structural method: GS-FLCN. For each problem, both algorithms were run from the same initial data and the algorithmic behavior and results were contrasted.

5.1. Example I: Model of a Distillation Column. The first example corresponds to the section of an ammonia synthesis plant where ammonia is purified by distillation. The corresponding flow diagram is shown in Figure 9. The main processing unit is a distillation column (D1) and the model includes the following six additional items of equipment: a mixer (MIX3), two dividers (DIV1 and DIV2) and three heat exchangers (CHX1, DRB1, and RFHX2). The steady-state mathematical model of this section is made up of 104 nonlinear algebraic equations with an initial sensor configuration composed of 25 measurements and 85 unmeasured variables.

The results from the observability analyses carried out with both methods are schematized in Table 1. It is

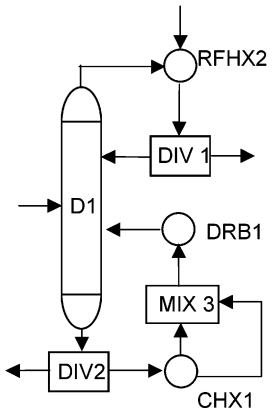


Figure 9. Flow diagram for the ammonia distillation column.

Table 1. Comparative Results from the Observability Analysis for Example I

dimension block size	104 × 100	
	GS-FLCN	direct method
1	55	44
2	1	1
5		1
6	1	
12		1
pav (%)	74	74
run times (min)	1:53	0:03

interesting to note that this is a problem of small size. As a consequence, it was possible to execute GS-FLCN thoroughly, without employing BFs. As a result, GS-FLCN could explore the complete search space, thus working at its very best in regards to robustness.

Both methods succeeded in determining the 63 observable variables in the problem. Therefore, the percentage of assigned variables (pav) was the same. Nevertheless, the configuration of the assignment subsets was different.

In general, GS-FLCN tends to yield smaller blocks than the direct method because of the strategy employed to look for the assignment subsets. GS-FLCN starts seeking for blocks of size n only after the exploration in search for blocks of size $n - 1$ has been completed. Besides, whenever the method locates a block of size n, with $n > 1$, the block is decoupled and the exploration starts all over again from the very beginning, i.e., trying to detect subsets of size 1. This policy guarantees minimum-size sets provided no techniques that limit the search space, such as BFs, are employed. This important feature of GS-FLCN is attractive because it is easier to detect unallowable assignment subsets on small blocks and it is also simpler to incorporate constraints. As to the run times, the direct method behaved much better. However, this problem is too small to be interesting for time comparisons.

According to these results, GS-FLCN is more robust (with respect to the detection of minimum-size blocks) than the direct method, even though both algorithms achieve the same efficacy level (i.e., the same amount of observable variables was detected). It can be concluded that when GS-FLCN is employed without resorting to techniques that prune the search space, the algorithm yields a classification whose granularity is either equal to or finer than the one obtained through the direct method. However, it is important to remark that this is a small problem with very few units. The mathematical models for wider sectors or industrial plants become significantly more complex, and their size increases dramatically. As a result, it is not always feasible to run GS-FLCN in its pure form, and the employment of acceleration strategies, such as the BFs, becomes mandatory.

Table 2. Results from the Observability Analysis for Example II

dimension block size	557 × 513	
	GS-FLCN	direct method
1	113	117
2	2	
5		2
6	2	2
7		1
8		1
9		1
pav (%)	25	32
run times (min)	35:12	0:14

5.2. Example II: Ammonia Synthesis Plant. The second example corresponds to an ammonia synthesis plant designed by Bike.[25] The mathematical model chosen to represent the plant consisted of 557 equations and 587 variables. The analysis was started from a basic configuration with a few instruments: only 74 variables were set as measured, with the rest being initially defined as unmeasured.

This is a medium-size problem that required the eventual use of BFs to reduce computing times when running GS-FLCN. In view of the fact that run times grow as the search depth augments, the search space was not reduced for the lower levels, i.e., where blocks of sizes between 1 and 6 were being looked for. From level 7 onward, different BFs were employed in order to reduce the space dimensions gradually. The higher the level, the stricter the pruning that had to be imposed in order to keep times within reasonable bounds.

Table 2 summarizes the results yielded by both GS-FLCN and the direct method. In contrast with case study I, the pav achieved through the direct method is slightly greater than the one obtained with the direct method. All of the variables assigned by GS-FLCN were also classified as observable by the direct method, which could also detect 34 other observable variables that GS-FLCN never individualized. This difference in the sets of assigned variables arose because GS-FLCN could not explore the entire search space when trying to locate subsets of size 7 or greater. As a result of the pruning strategy, some valid assignment subsets were not detected by the combinatorial algorithm.

The impact of failing to locate a block of a given size often propagates to the lower levels because the removal of an assignment subset of size n, for $n > 1$, frequently triggers the subsequent decoupling of smaller blocks. This effect manifested itself in this example. In Table 2, it can be observed that the direct method found four additional blocks of size 1, which could not be detected by GS-FLCN even though the BFs never impose bounds on the search for 1 × 1 blocks. Because the combinatorial algorithm could not find the bigger assignment subsets (of sizes 7−9), it never found those four smaller blocks.

Finally, in regards to execution times, the direct method proved to be significantly more efficient than GS-FLCN, with the times differing by several orders of magnitude. This gap can be attributed to the combinatorial nature of GS-FLCN, whose run times grow exponentially with both problem size and search depth.

Table 3. Comparative Results from the Observability Analysis for Example III

dimension block size	1830 × 1600	
	GS-FLCN	direct method
1	292	807
2	2	4
3	1	
4	1	1
5	1	
6	2	6
7		1
11		1
16		2
22		1
27		1
35		2
39		1
pav (%)	20	66
run times (min)	59:23	0:42

In brief, the direct method behaved notoriously better for this example because it was more efficacious and efficient than GS-FLCN, successfully assigning 7% more variables. As the size of occurrence submatrix **N** increased, it was necessary to employ BFs to speed up GS-FLCN, and this action was detrimental to the method's capacity to detect assignment subsets.

5.3. Example III: Ethane Plant. The last case under study corresponds to the real ethane plant presented in a paper by Ponzoni et al.[8] The rigorous mathematical model employed for this analysis contained 1830 equations with 332 measurements and 1600 unmeasured variables. This is the biggest example presented here, but it is important to mention that huge problems of much greater size are frequently encountered in engineering practice.

Because of the size of this problem, BFs were employed to accelerate GS-FLCN, just like in the previous example, though more severe pruning was required in this case in order to keep computing times reasonably low. Complete exploration of the graph was allowed up to depth 5. From level 6 onward, the search space was gradually bounded, with the strictest bound being set for levels above 8.

The results obtained after several global iterations with both methods are reported in Table 3. The last row of the table shows that the direct method could assign 66% of the unmeasured variables, while GS-FLCN only classified 20% as observable. This significant loss in efficacy exhibited by the combinatorial algorithm can be attributed to two main causes. The most evident reason is associated with the strict bounding on the search space required for a big problem that notoriously limited the exploration capacity of the method. Then, the method obviously failed to detect a great amount of valid blocks.

The second reason behind GS-FLCN's unsatisfactory performance is associated with the physical features that characterize this industrial process. Most of the streams flowing in the plant contain 12 chemical components, which favors the appearance of big assignment subsets that are typically formed around the global and component mass balances. In general, it can be stated that the greater the number of compounds considered in the mathematical model, the bigger those assignment subsets will be. This fact obviously affects GS-FLCN in a negative way because the exploration of the deeper levels associated with those subsets is severely bounded to limit execution times. Consequently, some of those subsets may remain hidden, as was the case in this problem where the largest blocks detected by the direct method (see Table 3) could never be found by means of GS-FLCN. Moreover, just like for example II, the computing time required by the combinatorial technique was notoriously higher, this being another strong advantage of the direct method.

The main conclusion that can be drawn from the analysis of this example is the fact that the behavior of the direct method will be significantly better for big-size industrial plants, where the formation of large assignment subsets are expected. Therefore, we strongly recommend the use of the proposed technique in those cases.

6. Conclusions

A direct method for observability analysis was presented in this paper. In contrast with other structural methods that had been proposed to treat complex nonlinear models, this technique does not carry out combinatorial searches, like GS-FLCN.[8] Moreover, it does not employ heuristic rules, like CDHG.[26]

The proposed method is a two-stage procedure based on graph decompositions that rearranges the occurrence matrix associated with the unmeasured variables by means of bigraphs and digraphs. The algorithmic core is constituted of a new node classification derived from Dulmage and Mendelsohn's technique for the decomposition of structurally singular matrices. A three-step strategy was designed for the identification and analysis of forbidden subsets. It comprises an allowability test, a subset reassignment procedure, and a bigraph reduction. The allowability test follows the same philosophy originally designed for the GS-FLCN and CDHG algorithms. In turn, the reassignment step aims at avoiding the repeated appearance of forbidden subsets by adequately permuting rows that correspond to assigned and redundant equations. Finally, the reduction phase makes it possible to build alternative maximum matchings either when the reassignment has not been successful or when it is not applicable.

The direct method was compared with GS-FLCN for three sample problems in order to assess its performance for the observability analysis of industrial plants. The results were extremely satisfactory because the direct method always managed to assign an equal or often higher amount of unmeasured variables. Although it was observed that GS-FLCN may yield a decomposition of finer granularity for small problems, it was stated that the direct method becomes more efficacious as problems grow in size and complexity. Besides, it also proved to be remarkably more efficient than GS-FLCN.

As to the scope of this development, it is interesting to note that the direct method constitutes a new strong matrix-partitioning tool, whatever the field of application. The technique is powerful because it can be applied to any matrices, regardless of their structural pattern. It is also original because no algorithms that allow the addition of block constraints in order to guide the search according to specific criteria could be found in the literature. Besides, the method is extremely flexible for it is possible to generate various reorderings for the same problem until a satisfactory one has been found.

Appendix: Algorithms

A1. Detection of a Single Strong Component in a Digraph (SSC).

Input: i, v, $\mathcal{N}$, $\mathcal{E}$, P
Input/Output: j, $\mathcal{CC}$

DFI(v) ← i.
$Q(v)$ ← DFI(v).
i ← i + 1.
Put v in P.
Stack(v) ← true.
For each node v' adjacent to v **do**:
  If DFI(v') = 0
    then
      SC(i,v',DFI,$\mathcal{N}$,$\mathcal{E}$,P,j,$\mathcal{CC}$).
      $Q(v)$ ← min ($Q(v)$, $Q(v')$).
    else
      If DFI(v') < DFI(v) **and** Stack(v')
        then
          $Q(v)$ ← min ($Q(v)$, DFI(v')).
      End if
  End if
End for
If $Q(v)$ = DFI(v)
  then
    Unstack all elements from P until v is reached.
    Store the unstacked elements in $\mathcal{CC}(j)$, including v.
    Unstack v.
    Set Stack(u) ← false, for all nodes in $\mathcal{CC}(j)$.
    j ← j + 1.
{*the elements stored in* $\mathcal{CC}(j)$ *constitute the jth strong component of the graph whose root is* v}
  End if
  End of Algorithm

A2. Detection of All of the Strong Components in a Digraph (SC).

Input: $\mathcal{N}$, $\mathcal{E}$
Input/Output: $\mathcal{CC}$

i ← 1.
j ← 1.
Empty P.
Empty $\mathcal{CC}$.

For all $v \in \mathcal{N}$ **do**:
  DFI(v) ← 0.
  Stack(v) ← false.
End do
While there is some u, so that DFI(u) = 0 **do**:
  SSC(i,u,DFI,$\mathcal{N}$,$\mathcal{E}$,P,j,$\mathcal{CC}$).
End while
End of Algorithm

A3. Maximum Matching in a Bigraph (MM).

Input: $\mathcal{R}$, C, $\mathcal{E}$
Output: $\mathcal{P}_m$

$\mathcal{P}_m$ ← ∅.
C_U ← ∅.
% Building an initial matching
For each node $c \in \mathcal{C}$ **do**:
  Match c with the first unmatched node r, so that $r \in \mathcal{R}$.
  If node r does not exist, **then** $\mathcal{O}$ ← $\mathcal{O} \cap \{c\}$
end do
% Looking for augmenting paths
$\mathcal{O}_\mathcal{N}$ ← ∅.
Repeat
  Search for an augmenting path $\mathcal{A}u$ from c, only visiting those nodes in R that have not been visited before during that step.
  Label all nodes that are reached as "visited".
  If an augmenting path $\mathcal{A}u$ has been found
    then
      Augment $\mathcal{P}_m$ with $\mathcal{A}u$.
    else
      Include c in $\mathcal{O}_\mathcal{N}$.
    End if
    $\mathcal{O}$ ← $\mathcal{O}_\mathcal{N}$.
    $\mathcal{O}_\mathcal{N}$ ← ∅.
Until no augmenting paths are found in the loop.
End of Algorithm

A4. Direct Method for the Classification of Unmeasured Variables.

Input: **N** (occurrence matrix) and **R** (initial constraints)
Output: **N** in BlTF

Stage 0. Initialization.
Build the bigraph $\boldsymbol{G}(\mathbf{N}) = (\mathcal{R},\mathcal{C},\mathcal{E})$ associated with **N**.

Stage 1. Coarse-Grain Decomposition.
1.1. Obtain a maximum matching $\mathcal{P}_m$ from $\boldsymbol{G}(\mathbf{N})$.
1.2. Classify the rows into $\mathcal{SR}1$, $\mathcal{SR}2$, $\mathcal{VR}$, and $\mathcal{HR}$ and the columns into $\mathcal{SC}1$, $\mathcal{SC}2$, and $\mathcal{HC}$ on the basis of $\mathcal{P}_m$.
1.3. For each *special row*, classify it as
  $\mathcal{VR}$, if all of its edges lead to columns in $\mathcal{SC}1$ or $\mathcal{SC}2$.
  $\mathcal{SR}1$, if it is connected to only one column in $\mathcal{HC}$ and it is not connected to any column in $\mathcal{SC}2$ (if this is the case, this column, together with the special row, constitutes a 1 × 1 block).
  $\mathcal{SR}2$, if it is connected to only one column in $\mathcal{HC}$ and at least one column in $\mathcal{SC}2$ (if this is the case, this column, together with the special row, constitutes a 1 × 1 block).
  $\mathcal{HR}$, if it is connected to more than one column in $\mathcal{HC}$.

Stage 2. Fine-Grain Decomposition.
2.1. Associate the digraph $\boldsymbol{G}(\mathbf{N}_1) = (\mathcal{V},\mathcal{E})$ with matrix $\mathbf{N}_1$, which corresponds to block ($\mathcal{SR}1$, $\mathcal{SC}1$).
2.2. Decompose $\boldsymbol{G}(\mathbf{N}_1)$ into its strong components $\mathbf{N}_{11}$, $\mathbf{N}_{12}$, ..., $\mathbf{N}_{1q}$ (each $\mathbf{N}_{1i}$ corresponds to a diagonal block).
2.3. Associate the digraph $\boldsymbol{G}(\mathbf{N}_2) = (\mathcal{V},\mathcal{E})$ with matrix $\mathbf{N}_2$, which corresponds to block ($\mathcal{SR}2$, $\mathcal{SC}2$).
2.4. Decompose $\boldsymbol{G}(\mathbf{N}_2)$ into its strong components $\mathbf{N}_{21}$, $\mathbf{N}_{22}$, ..., $\mathbf{N}_{2p}$ (each $\mathbf{N}_{2i}$ corresponds to a diagonal block).

Stage 3. Allowability Test.
3.1. For each strong component $\mathbf{N}_{1i}$,
  3.1.1. Check that $\mathbf{N}_{1i}$ does not belong to the set of constraints **R**.
  3.1.2. If $\mathbf{N}_{1i}$ is forbidden,
    go to stage 4.
  3.1.3. If the reassignment was successful,
    return to stage 2.
  Otherwise, go to stage 5 for a reduction to $\mathbf{N}_{1i}$.
3.2. For each strong component $\mathbf{N}_{2i}$,
  3.2.1. Check that $\mathbf{N}_{2i}$ does not belong to the set of constraints **R**.
  3.2.2. If $\mathbf{N}_{2i}$ is forbidden,
    go to stage 5 for a reduction to $\mathbf{N}_{2i}$.
3.3. Go to stage 6.

Stage 4. Reassignment of a Strong Component $\mathbf{N}_{1i}$.

4.1. Look for an edge (r, c), where $r \in \mathcal{VR}$, c belongs to the columns in $\mathbf{N}_{1i}$, $(k, c) \in \mathcal{P}_m$, and k has not been reassigned by r before.

4.2. If such an edge exists, then the reassignment is possible, and the following steps must be carried out:

4.2.1. Remove edge (k, c) from $\mathcal{P}_m$, where k is one of the rows in $\mathbf{N}_{1i}$.

4.2.2. Add (r, c) to $\mathcal{P}_m$; remove k from $\mathcal{SR}1$ and $\mathbf{N}_{1i}$.

4.2.3. Add r to $\mathcal{SR}1$ and $\mathbf{N}_{1i}$; remove r from $\mathcal{VR}$.

4.2.4. Add k to $\mathcal{VR}$.

Otherwise, the reassignment is impossible.

4.3. Go back to step 3.1.3.

Stage 5. Reduction of Bigraph $G(\mathbf{N})$ to the Strong Component $\mathbf{N}_{ij}$.

5.1. Remove all of the rows and columns corresponding to the strong components preceding $\mathbf{N}_{ij}$, i.e., all of the $\mathbf{N}_{kl}$ for $k < i$ or $k = i$ and $l < j$, from the bigraph and incorporate them to the solution.

5.2. Choose a row from $\mathbf{N}_{ij}$, which had not been selected before, as the *special row* and remove it from $\boldsymbol{G}(\mathbf{N})$.

5.3. If all of the rows in $\mathbf{N}_{ij}$ had been chosen as the *special row* before, choose two *special rows* among the rows in $\mathbf{N}_{ij}$, disregarding whether they had been selected previously.

5.4. Return to stage 1.

Stage 6. Reordering.

6.1. Rearrange $\mathbf{N}$ as follows:

$[\mathbf{N}_{11}, \mathbf{N}_{12}, ..., \mathbf{N}_{1p}, \mathbf{N}_{21}, \mathbf{N}_{22}, ..., \mathbf{N}_{2q}, (\mathcal{VR}, \mathcal{SC}1), (\mathcal{HR}, \mathcal{HC})]$

6.2. End of Algorithm

Literature Cited

(1) Vaclavek, V.; Loucka, M. Selection of Measurements Necessary to Achieve Multicomponent Mass Balances in Chemical Plants. *Chem. Eng. Sci.* **1976**, *31*, 1199−1205.

(2) Kretsovalis, A.; Mah, R. S. H. Observability and Redundancy Classification in Generalized Process Networks−II. Algorithms. *Comput. Chem. Eng.* **1988**, *12*, 689−703.

(3) Meyer, M.; Koehret, B.; Enjalbert, M. Data Reconciliation on Multicomponent Network Process. *Comput. Chem. Eng.* **1993**, *17*, 807−817.

(4) Crowe, C. M. Observability and Redundancy of Process Data for Steady-State Reconciliation. *Chem. Eng. Sci.* **1989**, *44*, 2909−2917.

(5) Madron, F. *Process Plant Performance. Measurement and Data Processing for Optimization and Retrofits*; Ellis Horwood Ltd.: Chichester, England, 1992.

(6) Romagnoli, J. A.; Stephanopoulos, G. On the Rectification of Measurement Errors for Complex Chemical Plants. *Chem. Eng. Sci.* **1980**, *35*, 1067−1081.

(7) Joris, P.; Kalitventzeff, B. Process Measurement Analysis and Validation. *XVIII Congress on the Use of Computers in Chemical Engineering*, CEF'87, Giardini Naxos, Italy, Apr 1987; pp 41−46.

(8) Ponzoni, I.; Sánchez, M. C.; Brignole, N. B. A New Structural Algorithm for Observability Classification. *Ind. Eng. Chem. Res.* **1999**, *38*, 3027−3035.

(9) Ponzoni, I.; Vazquez, G. E.; Sánchez, M. C.; Brignole, N. B. Parallel Observability Analysis on Networks of Workstations. *Comput. Chem. Eng.* **2001**, *25*, 997−1002.

(10) Gibbons, A. *Algorithmic Graph Theory*; Cambridge University Press: Newcastle, U.K., 1994.

(11) Balakrishnan, V. K. *Graph Theory*; McGraw-Hill: New York, 1997.

(12) Johnsonbaugh, R. *Discrete Mathematics*; Prentice Hall: Englewood, Cliffs, NJ, 1997.

(13) Asratian, A. S.; Denley, T. M. J.; Häggkvist, R. *Bipartite Graphs and Their Applications*; Cambridge University Press: Cambridge, U.K., 1998.

(14) Karpinski, M.; Rytter, W. *Fast Parallel Algorithms for Graph Matching Problems*; Oxford University Press: Oxford, England, 1998.

(15) Tarjan, R. E. Depth-First Search and Linear Graph Algorithms. *SIAM J. Comput.* **1972**, *1*, 146−160.

(16) Gustavson, F. G. Finding the Block Lower-Triangular Form of a Sparse Matrix. In *Sparse Matrix Computations*; Bunch, J. R., Rose, D. J., Eds.; Academic Press: London, U.K., 1976.

(17) Duff, I. S.; Reid, J. K. An Implementation of Tarjan's Algorithm for the Block Triangularization of a Matrix. *ACM Trans. Math. Software* **1978**, *4*, 137−147.

(18) Dulmage, A. L.; Mendelsohn, N. S. Coverings of Bipartite Graphs. *Can. J. Math.* **1958**, *10*, 517−534.

(19) Dulmage, A. L.; Mendelsohn, N. S. Two Algorithms for Bipartite Graphs. *J. Soc. Ind. Appl. Math.* **1963**, *11*, 183−194.

(20) Pothen, A.; Fan, C.-J. Computing the Block-Triangular Form of a Sparse Matrix. *ACM Trans. Math. Software* **1990**, *16*, 303−324.

(21) Ponzoni, I.; Sánchez, M. C.; Brignole, N. B. An Efficient Implementation of the First Least-Connected Node Algorithm. *Mec. Comput.* **1997**, *XVIII*, 629−638.

(22) Ponzoni, I.; Sánchez, M. C.; Brignole, N. B. Permutación de Matrices Ralas a una FTB Inferior Específica mediante Descomposición de Grafos. Workshop WAIT'97, Buenos Aires, Argentina, 1997.

(23) Ponzoni, I.; Sánchez, M. C.; Brignole, N. B. Permutation of Sparse Matrices to a specific lower BTF using Graph Decompositions. *Electron. J. Argent. Soc. Inf. Oper. Res.* **1998**, *1*, 76−87.

(24) Hopcroft, J. E.; Karp, R. M. An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM J. Comput* **1973**, *2*, 225−231.

(25) Bike, S. *Design of an Ammonia Synthesis Plant*; CACHE Case Study Report; Department of Chemical Engineering, Carnegie Mellon University: Pittsburgh, PA, 1985.

(26) Ponzoni, I.; Sánchez, M. C.; Brignole, N. B. CDHG: a New Partitioning Algorithm based on the Detection of Cycles in Hypergraphs. *Lat. Am. Appl. Res.* **1998**, *28*, 31−36.

Received for review January 14, 2003
Revised manuscript received September 29, 2003
Accepted October 29, 2003

IE0300326