Hierarchical representation of complex systems for supporting human decision making

S. Gentil, J. Montmain

Hierarchical representation of complex systems for supporting human decision making

S. Gentil[a], J. Montmain[b,*,1]

[a]Laboratoire d'Automatique de Grenoble, CNRS-INPG-UJF, BP 46 38402 St Martin d'Hères, France

[b]Commissariat à l'Energie Atomique (CEA), URC EMA-CEA, Site EERIE, Parc Scientifique G. Besse 30035 Nîmes, Cedex, France

Abstract

The work presented in this paper is devoted to intelligent on-line supervision tools. In the proposed approach, the human operator remains in the decision loop, at the highest level, and acts on the process. To help operators make decisions, process knowledge is represented with a model whose complexity can be adapted on line to the operation needs at the request of the operator. The model thus helps to focus only on the phenomena that are relevant at a given time. To give the model explanatory capacity, it is represented as a causal directed graph, and allows the representation of temporal phenomena, which is fundamental for dynamic monitoring. A hierarchical representation of the functional properties of the process is proposed. The conception of a hierarchy of causal models with a top-down analysis is discussed. Path algebra is used to construct a higher-level graph on-line at the request of the operator from the most detailed graph, while conserving the semantics of the latter. No intermediate level is defined a priori; only the highest and lowest level graphs are fixed: the others are constructed dynamically. Finally, a study of how graphs can convey information on the dynamics of the process for approximate temporal reasoning that is largely sufficient for supervision purposes is analyzed. An example of a causal graph hierarchy for a nuclear process illustrates the method. As a final point, the use of such causal graphs in advanced industrial supervision tools is considered.

Keywords: Supervision; Operation support; Causal modeling; Graph theory; Hierarchical modeling

1. Introduction

The human cognitive modes of comprehension, perception, representation and decision making have been studied for a long time. Analyses have been applied in particular to situations in which human operators are controlling and/or supervising a technical plant. Their global objective is first to maintain a safe operation and then to optimize some production criteria related to product quality, energy saving, production speed, etc. The information they handle is mainly obtained with an on-line data acquisition system connected to process sensors. They deal with this information through their knowledge of the process related to physical components, their functions, the behavior of their characteristic variables and the relations among them.

The pioneering work by Rasmussen [1] has shown the many tasks that an operator is supposed to execute. Such activities include data monitoring and information seeking, pattern recognition, diagnosing, planning and acting on the system. The various control strategies used by the operator (reversion to manual mode, degraded operation, automatic shutdown, etc.) are clearly task-dependent [2]. Generally, in routine situations, the operator simply monitors a few variables. Operator intervention is necessary in the event of a system malfunction or a change in the operating mode. Human activity may thus be broken down into three steps: perception and cognition followed either by the formulation of a strategy to keep the system operating in a faulty situation or by system shutdown [3]. Rasmussen's model is organized around three types of behavior: skill-based (direct mapping from observation to situation); rule-based (stereotyped procedure to execute a task [4]); knowledge-based

E-*mail addresses:* sylviane.gentil@inpg.fr (S. Gentil), jacky.montmain@ema.fr (J. Montmain).

(decision making on a new situation). Hoc [5] revised this model to take into account the operators' behavior in a dynamic environment and their capacity to anticipate the process's future evolution.

The analysis of a human operator's mental activity provides model that can be helpful in the design of new tools devoted to intelligent on-line supervision [6]. Excluding humans from supervisory decision making and replacing operators with a fully computerized system is not reasonable. Humans are still necessary to do some high level cognitive tasks [7], and supervisory systems should be designed to support them in their decision making tasks. Thus a cooperative problem-solving approach [8] is generally adopted, even for highly complex automated processes.

The current design strategy for supervision systems consists in presenting, in the most ergonomic way, much information about the process structure and the variables' time evolution. The operators can navigate from one interface view to another, seeking information about one or another sub-system/component. An intelligent supervisory system is mainly expected to structure this information, based on a model of the facility. To be useful, this model must be in full agreement with the operator's mental reasoning.

The supervision of complex systems has been an area where two research communities have been particularly active: the Artificial Intelligence community [9] and the Control Theory community [10]. The general objective of AI is to reproduce human reasoning and more generally any human cognitive mode of comprehension, perception, representation and decision making. It is thus human reasoning centered and has a lot of shortcomings with respect to signal processing and analysis of continuous dynamics that are still the basic foundations of many activities supported by the Digital Control and Monitoring System (DCMS) in control rooms. Conversely, Control Theory processes numerical data and algorithms in order to stabilize systems or optimize production. It is purely DCMS oriented: the numerical assessments of the DCMS are never accompanied by elucidations although the operator is still considered as the final decision maker in the control room.

It is well known that humans avoid calculations whenever possible. This has led the AI community to study qualitative reasoning about the topological properties and qualitative modeling of physical systems. Understandability is an important requirement for supervision and is generally opposed to accuracy. It may be sufficient in some cases only to know that a relation exists among variables or merely to specify the orders of magnitude of the phenomena.

Temporal reasoning is another fundamental process leading to decision making [11]. Transient system behavior is essential knowledge for managing plant production or safety. In case of malfunctioning, the operators must perceive new events that dynamically modify the process behavior. Temporal dynamics thus constitute an important factor that must be taken into consideration [12,13]. Control theory is focused on the numerical representation of dynamic physical phenomena, generally with differential equations. The characteristic of classical control models is that they are purely quantitative. Thus, a lot of time has to be spent on system identification and model parameter estimation because an accurate model is required for control purposes. If this has to be done at the facility scale, it may be dissuasive. However, the requirements are different for supervision purposes, and a rough idea of the time necessary for a variable to attain a steady state may be sufficient to make a decision.

Causality occupies a central position in human cognition [14]. The AI community has been working for a long time on representations of causality. Informal descriptions of real-world phenomena in the form *A causes B*, are very common. *B* can be predicted or explained using *A*. In particular, causal modeling, whether applied in the context of economic systems or qualitative physics, has been the subject of a famous debate [15,16]. Causal modeling enables a complicated process to be decomposed into elementary sub-models and is thus very suitable for complex system analysis. Causal models provide explanations of the behavior of the modeled system that are close to human reasoning, which is completely excluded by the purely numerical calculus that constitutes the basis of control theory. Causal descriptions are the source of various reasoning modes useful for supervision: *understanding, predicting, diagnosing and action advice* [17]. Causality plays an essential role in human decision making by providing a basis for choosing that action that is likely to lead to a desired result. Diagnosis, an important supervision aspect, is also typically a causal process because it consists in designating the faulty components that have caused, and can explain, the observed malfunctions. In this respect, the objective of the use of a causal model is to deal with the combinatorial explosion that arises with model-based diagnosis approaches [18–21].

Causal interpretation, in the sense of temporal precedence of events, is a fundamental tool in enhancing the intelligent behavior of the operator. Thus qualitative, causal and temporal knowledge appear as good features for a model dedicated to cooperative supervision. An examination of the contributions of AI and Control Theory could lead to the conclusion that reasoning and computing are in opposition. The modeling method used in this paper combines them and is presented in detail in [22]. It relies on both a qualitative causal representation of the process and on quantitative elementary models. It has been inspired by Artificial Intelligence for the causal modeling of physical systems. But it takes advantage of Control Theory in the way the process dynamics are taken into account using relations between variables that manage time explicitly.

Nevertheless, the complexity of a single model describing an entire facility may make it incomprehensible to the operator. On traditional industrial control consoles,

only one level of resolution is available, and this must be the most detailed level required in any situation. The operator must therefore perform a variety of reasoning tasks using the same detailed topological model of the entire facility. A good way to cope with control of complex systems would be to structure the information. One dimension for structuring is the level of abstraction in the representation, resulting in an abstraction hierarchy [23]. Hierarchical decomposition of models is crucial for managing complexity by refining model components into models of the same type as the lumped model. The models must be homogeneous to relate the conclusions drawn from different models and to generate a coherent description of the system behavior. Iwasaki [24] defines four axes of abstraction: structural, functional, temporal and quantitative. The model developed in this paper is functional, not structural, a choice that appears better suited to an overall vision of a process from the standpoint of long-term planning, reasoning about the evolution of physical phenomena, perception and interpretation of a malfunction.

This paper proposes an original multi-level process representation with varying levels of detail that is helpful to human decision making. The models at intermediate levels are automatically created using path algebra. This hierarchical representation completes the dynamical causal model presented in [22].

Section 2 describes the conception of a hierarchy of dynamic causal models, represented by graphs, and defines what is subsequently referred to as a graph hierarchy. A method is discussed for developing a top-down graph hierarchy by thorough analysis of the supervised process. Section 3 indicates a method for constructing a graph online, at the request of the operator, from the knowledge of the most detailed graph, while conserving the semantics of the latter. This method is based on the representation of a graph with a matrix and on path algebra. No intermediate level is defined a priori. Only the highest and lowest level graphs are predetermined. The others ones are constructed dynamically. During system operation, tasks can be formulated at any level. Finally, Section 4 shows how graphs can convey information on the dynamics of the process, for approximate temporal reasoning that is largely sufficient for supervision purposes. Section 5 provides an example of top-down construction of a causal graph hierarchy for a nuclear process.

2. Conception of causal graphs

A graph is a knowledge representation structure consisting of nodes interconnected by arcs. A graph is 'causal' if the semantics of the arcs represent the property of causality: an input node of an arc is one of the causes of the output node.

In the following discussion, the nodes represent process variables, i.e. physical quantities that are meaningful to the operators of process operation; the arcs represent the functions relating the variables. All the variables are not necessarily measurable; some may be reconstructed from other measurements or represent concepts useful for supervision. Nevertheless, complex processes are generally highly instrumented for safety reasons, and many variables can be displayed in the operator interface and used for process monitoring.

In addition to causality, represented by the direction of the arc, each arc may be assigned some degree of semantics. For example, the first studies of alarm processing [25–28] use signed influence indicators: the arc is marked $+$ or $-$, depending on whether the output variable evolves directly or inversely with the input variable. The gain amplitude may be used to relate orders of magnitude of variable variations [13,29]. Information on the temporal dynamics among variables may also be used [22,30]: the arc then represents the transfer function between the input variable and the output variable, and the causal graph becomes the equivalent of the process block diagram (refer to Section 4).

2.1. Definition of a causal graph hierarchy

To define the type of hierarchy that should be used to represent the causal graphs, careful consideration must be given to their assumed usage by the human operators. Throughout the supervision task, the operators monitor the process by regularly observing a relatively small set of variables that defines the high-level graph in the hierarchy. When the operators feel the situation is no longer normal, they will focus on a particular subsystem to verify hypotheses or identify means of action if the problem is correctly understood. The operators may also simply display more variables to understand the situation and follow its evolution.

A graph G_2 at a lower hierarchical level than graph G_1 must therefore contain the nodes of G_1 together with a number of other nodes used either to take additional phenomena into account or to detail certain functions by modeling internal events. Under no circumstances could any nodes of G_1 not appear in G_2: it would be difficult to imagine an operator attempting to understand a situation or to diagnose a fault condition and being presented with fewer variables, or seeking a better means of action to remedy a malfunction and seeing certain means of action displayed on the high-level graph but absent from a more detailed graph. Consequently, the detailed graph may contain new sources (e.g. a regulation set point), new sinks (e.g. additional measurements), or intermediate variables (for a more detailed perception of some phenomena). In the last case, an arc in the higher-level graph is broken down into a path in the lower-level graph.

If the graph arcs represent transfer functions, this hierarchical breakdown procedure consists in revealing internal variables by breaking down a complex transfer function into a product of elementary transfer functions.

These internal variables must be meaningful to the human operator. The same situation arises if the arcs only contain gain information or signed influence indicators.

The level of a graph is thus defined by a set of variables:

$$N_i = \{V_1, \ldots, V_{ni}\} \quad n_i = \mathrm{card}(N_i)$$

corresponding to the nodes of graph G_i. A graph G_j represents a lower level than G_i if:

$$N_j = N_i \cup \{V'_1, \ldots, V'_{kj}\} \quad n_j > n_i$$

2.2. Conception of a causal graph hierarchy

A top-down approach is adopted to construct the causal graph hierarchy a priori. The construction is initialized by graph G_0, including the variables used for supervision when the process situation is normal; n_0 should normally be about 10. They are the variables that are monitored during the normal overall operation of the facility, reflecting global mass balances or energy balances for example.

Additional variables are then introduced, corresponding to additional measurements of a single quantity (e.g. monitoring of the temperature variation along a pipe). A complex function may also be broken down into elementary functions (for example, a motor may be represented by an arc between the power supply voltage and the speed, or the current intensity may be added as an intermediate variable); an arc is thus replaced by a path. The representation may also focus on a particular function, such as regulation. In this case, at a high level of abstraction, it is natural to assume that a regulated variable is equal to its set point and to represent both by a single node. In order to monitor transients or diagnose regulation malfunctions, the arcs relating the set point and the disturbances to the regulated quantity and to the action must be developed.

In this way, by adding an increasing number of variables, an increasingly detailed graph is obtained step by step until the graph contains all the variables considered potentially useful for supervision purposes. This graph may, of course, be very complex, since it contains all the information concerning the process. However, it is constructed off-line, once and for all; this corresponds to the step of knowledge extraction and representation necessary for the development of any intelligent system. The hierarchical decomposition method was tested on a nuclear fuel reprocessing plant. Results are presented in Section 5.

The situation is very different when this graph is used on-line by human operators. The operators generally begin with graph G_0 and wish to focus on a particular subsystem according to the context: i.e. either display additional measurements or detail a function. It is thus impossible to specify any a priori relevant hierarchical levels. Construction of the detailed graph is context-dependent at the request of the human operator. This approach is at the heart of human-machine cooperation and attempts to elicit intelligent behavior from the operator. The supervisory system exists only to provide the model requested by the human operator and is thus an example of 'integrated human-computer interaction' as defined by Johanssen [8]. The required graph is constructed from the definition of its hierarchical level (current graph variables and additional operator-requested variables), and is obtained on-line from the lowest-level (most detailed) graph.

Considering the potential number of combinations that could be requested by the operator, it is not practicable to store all the possible graphs; instead, it is preferable to develop a method for on-line construction of a synthetic graph from the detailed graph. It is easy to delete paths from non-relevant sources or leading to non-relevant sinks. It is more complicated to reduce the complexity of the graph between relevant variables, i.e. to join arcs so that a path between variables that the operator wishes to display becomes an arc, without the intermediate variables.

Section 3 describes in detail the procedure for finding any given path on-line in a graph by joining its arc semantic information. Section 4 details a more complex situation in which the semantics correspond to the dynamics among variables, represented by transfer functions. Consequently, a method is necessary for approximating the dynamics of a higher-level graph in such a way that the models supported by the arcs are not too complex and do not require excessive simulation time.

3. On-line construction of a causal graph hierarchy

In order to construct a graph on-line, it is necessary to identify the paths of the low-level graph that will become arcs in a higher-level graph. This process must also ensure that no relevant information from the detailed path is lost.

3.1. Path-seeking in a graph

Path algebra provides an elegant solution to many problems involving graphs [31]. This method uses a matrix representation of the graphs along with suitable algebraic structures. The set S of these matrices together with a sum $\oplus$ and a product $\otimes$, is a dioid $(S, \oplus, \otimes)$ or half-ring. The dioid $\oplus$ and $\otimes$ laws have the following properties:

- the addition $\oplus$ provides S with a commutative monoid structure (closure, associativity and commutativity). There is a neutral element ε or 'null element'.

$$\forall a, b, c \in S \begin{cases} a \oplus b \in S \\ a \oplus (b \oplus c) = (a \oplus b) \oplus c \\ a \oplus b = b \oplus a \\ a \oplus \varepsilon = a \end{cases} \tag{1}$$

- the product $\otimes$ provides S with a monoid structure (closure and associativity). There is a neutral element e or 'unity'.

$$\forall a,b,c \in S \begin{cases} a \otimes b \in S \\ (a \otimes b) \otimes c = a \otimes (b \otimes c) \\ a \otimes e = e \otimes a = a \end{cases} \quad (2)$$

- the product is distributive to the left and right with respect to the addition and accepts the null element as an absorbing element.

$$\forall a,b,c \in S \begin{cases} a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \\ (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c) \\ a \otimes \varepsilon = \varepsilon \otimes a = \varepsilon \end{cases} \quad (3)$$

This algebra provides an effective solution to many problems by defining the nature of the matrix elements and the additive and multiplicative laws. Section 3.1.1 shows that this approach can be used to find the number of paths of a given length between two graph variables by a simple matrix product. The paths can also be enumerated by assigning a name to each arc, as explained in Section 3.1.2. Section 3.1.3 discusses the limits of this construction, and Section 3.1.4 considers the approach required for loops.

3.1.1. Detecting paths in a graph

In order to clarify this discussion, consider the graph in Fig. 1. The graph is represented by the matrix of paths C_V where the variables are arranged in the order given by vector V (Eq. (4)). If only the causal relations among variables are relevant, it is unnecessary to specify the information supported by the arcs. The matrix elements contain Booleans.

$$C_V = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad V = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \quad (4)$$

The path matrix C_V contains all the directed paths of length 1 in the graph. The element of the ith row and jth column $C_V(i,j)$ is '1' if the directed arc between the jth variable and the ith variable exists; otherwise it is '0'. The nth power of matrix C_V must be calculated to obtain the paths of length n. The $\oplus$ and $\otimes$ operators are then the common matrix sum and product.

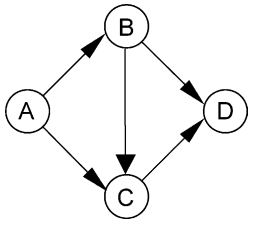


Fig. 1. A directed graph example.

Calculating the square of C_V shows that there are four paths of length 2, corresponding to the non-zero elements of C_V^2 $C_V^2(4,1) = 2$; shows that two paths of length 2 exist between A and D (see Fig. 1: one passes through B, the other through C). Similarly, calculating the cube and fourth power of C_V shows that there is one path of length 3 and no paths of length greater than or equal to 4 (Eq. (5)). A loop is identified simply when a '1' element appears on the diagonal of a matrix.

$$C_V^2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ \boxed{2} & 1 & 0 & 0 \end{bmatrix} \quad C_V^3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad C_V^4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (5)$$

The set of paths of any length is obtained by the limit of the sum C_V^* of all the powers of the path matrix C_V. If there are no loops in the graph, the calculation may be limited to the first N powers of C_V to obtain the existence of paths. In a graph with N nodes, there can be no paths with more than N arcs. Eq. (6) shows the set of paths for the graph in Fig. 1; for example, C_V^* (4,1) shows that there are a total of three paths between A and D.

$$C_V^* = \sum_{n=1}^{4} C_V^n = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \boxed{3} & 2 & 1 & 0 \end{bmatrix} \quad (6)$$

3.1.2. Enumerating the paths in a graph

Path algebra can be used not only to count the paths of a graph, but also to enumerate them. For this purpose, the arcs must be named (Fig. 2). The 'name' reflects the semantics assigned to the arcs; this may be a sign or a more complicated function. Enumeration, for each length of a given path, yields a character string corresponding to the aggregation of the arcs in the path. The graph in Fig. 2 is represented by matrix N_V when the variables are arranged in the order given by vector V (Eq. (7)). Matrix N_V contains all the paths of length 1 in the graph. As it gives the path names, it will be referred to as the matrix of names. For example, $N_V(2,1) = a$ represents the arc of

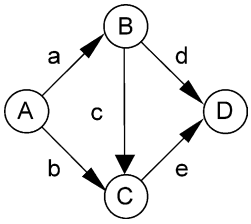


Fig. 2. A graph example with named arcs.

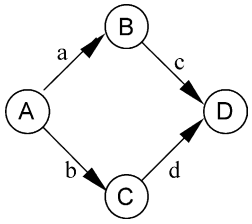


Fig. 3. A graph example with parallel paths.

semantic a from node A to node B.

$$N_V = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \boxed{a} & 0 & 0 & 0 \\ b & c & 0 & 0 \\ 0 & d & e & 0 \end{bmatrix} \quad V = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \tag{7}$$

The paths of length n are obtained by calculating the nth power of matrix N_V. The $\oplus$ and $\otimes$ operators then represent the union and concatenation of the character strings. It is important to note, however, that the concatenation is not commutative, as the arcs are directed. Union is represented in the following discussion by a simple + operator to simplify the notation. Thus, in the product of matrix $B=(b_{ij})$ by matrix $C=(c_{ij})$, the element of the resulting matrix A is $\sum_k b_{ik} \cdot c_{kj}$. Similarly, $N_V^n = N_V \otimes N_V^{n-1}$.

By calculating the square and cube of N_V, four paths containing two arcs and one path containing three arcs can be named (Eq. (8)); as noted above (Eq. (5)) there are no paths with four or more arcs. For example, the path of length 3 between A and D is ace.

The total number of paths is also obtained by the limit of the sum N_V^* of all the powers of matrix N_V (Eq. (9)). Element N_V^* (4,1) indicates three possible paths between A and D.

$$N_V^2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ ac & 0 & 0 & 0 \\ ad + be & ce & 0 & 0 \end{bmatrix} \tag{8}$$

$$N_V^3 = N_V \otimes N_V^2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \boxed{ace} & 0 & 0 & 0 \end{bmatrix}$$

$$N_V^* = \sum_{n=1}^{\infty} N_V^n = \begin{bmatrix} 0 & 0 & 0 & 0 \\ a & 0 & 0 & 0 \\ b + ac & c & 0 & 0 \\ \boxed{ad + be + ace} & d + ce & e & 0 \end{bmatrix} \tag{9}$$

This method is thus capable of finding the names of all the paths relating two variables in a graph, while respecting the direction of the arcs and thus the causality.

3.1.3. Limits of the method

Path algebra is highly effective in identifying the properties of a graph. It can quickly calculate the number of paths between two variables or detect the existence of loops by simple matrix multiplication. The limitations of this algebra appear when reducing a graph, i.e. selecting variables from the detailed graph that must be retained in the higher-level graph; this also identifies the variables that must be deleted. All the paths connecting two retained variables must then be identified.

Consider, for example, the simple graph in Fig. 3, showing two parallel paths connecting A to D passing through either B or C. The matrix representing the graph as well as its square and all possible paths is given by Eqs. (10) and (11). Note that there are no paths with lengths greater than 2: all the powers of N_V greater than 2 are zero.

$$N_V = \begin{bmatrix} 0 & 0 & 0 & 0 \\ a & 0 & 0 & 0 \\ b & 0 & 0 & 0 \\ 0 & c & d & 0 \end{bmatrix} \quad V = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \tag{10}$$

$$N_V^2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ ac + bd & 0 & 0 & 0 \end{bmatrix}$$

$$N_V^* = \begin{bmatrix} 0 & 0 & 0 & 0 \\ a & 0 & 0 & 0 \\ \boxed{b} & 0 & 0 & 0 \\ \boxed{ac + bd} & c & \boxed{d} & 0 \end{bmatrix} \tag{11}$$

Assume that only nodes A, B and D will be retained. It is not possible simply to delete the row and column corresponding to node C. The third row and the third column of matrix N_V^* indicate that there is a path from A to C and a path from C to D. The higher-level graph containing only A, B and D must therefore contain an arc connecting A and D. The problem is thus to determine the semantic content of that arc. Considering N_V^* (4,1) directly, there are two links between A and D. This element indicates not only path ac passing through B, but also path bd passing through C; only the second must be reduced. Calculating only the powers of

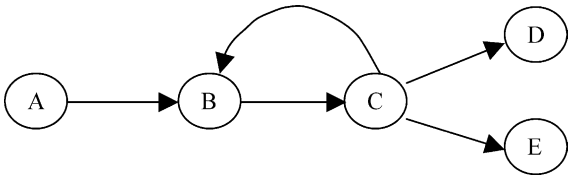


Fig. 4. A graph with a loop.

matrix N_V and matrix N_V^* is thus not sufficient to discriminate between paths *ac* and *bd*.

Reducing the graph therefore requires additional processing to discriminate between the influence of variables retained in the higher-level graph and the influence of variables that must be eliminated. An iterative solution to this problem is discussed in Section 3.2.

3.1.4. The case of loops

A graph is not limited to a set of paths interconnected in series or in parallel. The many coupling relations in an industrial process imply the existence of loops (Fig. 4).

One type of loop—the most common—is related to control loops. When a process is monitored from a very general standpoint, the set point and the regulated variable may be assumed identical. Conversely, to monitor transients or to diagnose a malfunction in the control loop, the arcs connecting the set point and the disturbances to the regulated quantity and the control signal must be developed. In this case, the change in the represented system properties is not merely the addition of detailed information on the physical variables; more fundamentally, information is added about the principles governing the lower level. The higher level represents the purpose of the system (a variable is equal to its set point); the lower level represents a change in the degree of abstraction, as mentioned in [23].

The relations between variables in a control loop are not naturally described by causal links because these variables influence each other instantaneously through paths with zero delay. In [17] a causal decomposition of regulation systems is obtained without the use of loops. In this decomposition, the set point variable and the disturbance variables are directly linked to the regulated variable and the control variable by suitable transfer functions. The latter variables evolve simultaneously following either a change in the set point value or a disturbance affecting the regulation. The decomposition of the regulation system is thus highly understandable.

The loops that raise real problems in reducing a graph are those describing the feedback of certain variables (e.g. material loops). The existence of such loops creates a serious difficulty—notably for diagnostic purposes—since a defect observed at one point in the loop may result from another defect at any other point in the loop. To ensure the operator is aware of this problem, it seems important to conserve the representation of a loop by a graph with at least two nodes. Deleting one of these nodes would imply defining an arc from a node to itself; such an arc would not clarify a causal explanation of the phenomena. When a loop of this type is detected, it must therefore not be reduced to fewer than two nodes. The nodes to be retained must be chosen according to the explanatory interest of the corresponding variables. For example, the nodes with the largest number of connections outside the loop could be retained.

The links between a loop and the rest of the process may be analyzed. The path matrix C_V (Eq. (12)) is used to count the number of links between *B* and *C* and the nodes outside the loop (Fig. 4). Calculating C_V^2 reveals the presence of loops by the two non-zero diagonal elements. Vector V' is introduced, in which only the elements corresponding to nodes outside the loop have non-zero values.

$$C_V = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad V = \begin{bmatrix} A \\ B \\ C \\ D \\ E \end{bmatrix} \quad V' = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \begin{matrix} \leftarrow A \\ \leftarrow B \\ \leftarrow C \\ \leftarrow D \\ \leftarrow E \end{matrix} \tag{12}$$

The product of C_V by V' is a vector containing the number of arcs entering a variable, coming neither from *B* nor from *C* (Eq. (13)), and thus outside the loop. The product of the transpose of C_V by V' is a similar vector with the number of outgoing arcs (Eq. (14)) unrelated to the loop. In this case, *B* has one link outside the loop, while *C* has two.

$$C_V \cdot V' = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow B \\ \leftarrow C \\ \\ \\ \end{matrix} \tag{13}$$

$$C_V^T \cdot V' = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \\ \leftarrow B \\ \leftarrow C \\ \\ \\ \end{matrix} \tag{14}$$

This simple method is capable of detecting links between the nodes of a loop and the nodes outside the loop. When reducing the graph, the two nodes with the largest number of links could be arbitrarily retained, unless the operators prefer to select nodes they consider as having the greatest explanatory potential.

The matrix representation of a graph allows loops to be detected. The graph as requested by the operator may thus be analyzed. If at least two nodes of the loop are to be retained, the graph may be constructed as indicated in Section 3.2. Conversely, if all the nodes of the detected loop (except perhaps only one of them) are to be eliminated, two

arbitrary nodes must be chosen and retained; possible selection criteria may be the number of links between nodes of the loop and those outside the loop, or the operator's designation.

3.2. Iterative reduction of a graph

Operators have chosen the list of variables defining the model they wish to consult. Representing the graph as a matrix and calculating the powers of the matrix are means of identifying any loops and assessing the feasibility of constructing the requested graph. After validating the level of the desired graph, all the extraneous nodes must be eliminated.

The elimination of a node from a graph is based on the product of the graph matrix and a matrix containing only the influence of the node to be eliminated. This matrix product is used to calculate a matrix in which the influence of an eliminated node is taken into account by the links between the remaining nodes.

Consider the example in Fig. 5 and its matrix representation G_{ABC} (Eq. (15)).

$$G_{ABC} = \begin{bmatrix} 0 & \boxed{0} & 0 \\ a & \boxed{0} & 0 \\ 0 & \boxed{b} & 0 \end{bmatrix} \quad V = \begin{bmatrix} A \\ B \\ C \end{bmatrix} \tag{15}$$

The objective is to delete node B. The influence of B on the rest of the graph is exerted via the outgoing arcs and is thus contained in the column corresponding to B. Matrix M_B is constructed by adding the second column of G_{ABC} to an identity matrix of the same dimensions as G_{ABC} (Eq. (16)).

$$M_B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & b & 1 \end{bmatrix} \tag{16}$$

$$M_B \otimes G_{ABC} = \begin{bmatrix} 0 & 0 & 0 \\ a & 0 & 0 \\ \boxed{ab} & b & 0 \end{bmatrix} \tag{17}$$

The left product of G_{ABC} by M_B yields a matrix that may be interpreted as representing the initial graph plus the paths of length 2 containing B. The example (Eq. (17)) represents an arc AC. B may then be eliminated by deleting the row and the column corresponding to its influence. This may be accomplished by left multiplying by the identity matrix in which the row corresponding to B has been deleted and by right multiplying by the identity matrix in which the column

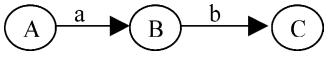


Fig. 5. A graph to reduce.

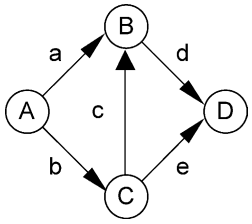


Fig. 6. B and C nodes elimination.

corresponding to B has been deleted. The resulting 2×2 matrix G_{AC} corresponds to the reduced graph (Eq. (18)).

$$G_{AC} = \begin{bmatrix} 0 & 0 \\ ab & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ a & 0 & 0 \\ ab & b & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{18}$$

When several nodes are to be eliminated from a graph, they must be deleted one at a time in an iterative process. Consider the example in Fig. 6; the matrix N_{ABCD} contains the names of the arcs (Eq. (19)).

$$N_{ABCD} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ a & 0 & c & 0 \\ b & 0 & 0 & 0 \\ 0 & d & e & 0 \end{bmatrix} \quad V = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \tag{19}$$

Let us assume that B and C are to be eliminated. The proper construction (Eq. (20)) can only be achieved by an iterative reduction, which ensures at each iteration that the resulting graph remains coherent with the lower-level graph by taking all the paths between nodes into account. The order in which B and C are eliminated is not important.

$$N_{AB} = \begin{bmatrix} 0 & 0 \\ ad + be + bcd & 0 \end{bmatrix} \tag{20}$$

Eq. (21) shows the matrix N_{ABD} resulting from the elimination of C in N_{ABCD}; Eq. (22) shows the matrix N_{ACD} resulting from the elimination of B in N_{ABCD}.

$$N_{ABD} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & c & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & e & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ a & 0 & c & 0 \\ b & 0 & 0 & 0 \\ 0 & d & e & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ a + bc & 0 & 0 \\ be & d & 0 \end{bmatrix} \tag{21}$$

$$N_{ACD} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & d & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ a & 0 & c & 0 \\ b & 0 & 0 & 0 \\ 0 & d & e & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 0 & 0 \\ b & 0 & 0 \\ ad & cd+e & 0 \end{bmatrix} \tag{22}$$

Eliminating B from N_{ABD} and eliminating C from N_{ACD} both yield the same matrix N_{AD} (Eq. (20)), as shown in Eqs. (23) and (24).

$$N_{AD} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & d & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ a+bc & 0 & 0 \\ be & d & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 0 \\ ad+be+bcd & 0 \end{bmatrix} \tag{23}$$

$$N_{AD} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & cd+e & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ b & 0 & 0 \\ ad & cd+e & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 0 \\ ad+be+bcd & 0 \end{bmatrix} \tag{24}$$

An iterative algebraic method for calculating the semantics of a high-level graph from those of a detailed graph has thus been defined. The level of detail of the model is defined by the variables used. This allows successive elimination of unnecessary variables from the detailed graph to obtain the high-level graph.

4. Assigning time parameters to arcs using transfer functions

The graph could be used by the operator to test a hypothesis concerning the origin of an observation or to verify the time required for a variable to respond to an action, i.e. as a simulation tool; the temporal dynamics then become an important parameter to take into consideration. However, precision is not the strong point of this simulator, since it is not used to solve control problems but rather supervision problems [32].

In the following discussion, the arcs of the detailed graph will be assigned temporal parameter values by means of very simple transfer functions to provide an approximate description of process behavior in the nominal operating mode without the need of a detailed physical model. The transfer functions may be obtained empirically from the orders of magnitude of the response times and delay times or, more precisely if necessary, by estimating the parameters from experimental records [33].

Three types of classic transfer functions are used here: the Strejc function (Eq. (25)) [34], the differentiator (Eq. (26)) and the integrator (Eq. (27)). The only parameters used are the gain g, the delay d, the time constant T and the order n.

$$F(s) = \frac{g \bullet e^{-d \bullet s}}{(1 + T \bullet s)^n} \quad n \in \{0, 1, 2, 3, 4\} \tag{25}$$

$$F(s) = \frac{g \bullet s \bullet e^{-d.s}}{(1 + T \bullet s)^n} \quad n \in \{0, 1, 2\} \tag{26}$$

$$F(s) = \frac{g \bullet e^{-d \bullet s}}{s} \tag{27}$$

Operators $\otimes_F$ and $\oplus_F$ must therefore be defined that are compatible with these functions. As they are described by four parameters (gain, time constant, delay and order), a direct combination (in the sense of conventional transfer function calculations) of only two of them would already involve eight parameters. This accumulation of parameters is contrary to the required simplicity of the model and does not ensure closure of the operations in the dioid. The complexity of the conventional sum and product operations on transfer functions must therefore be reduced without excessive effect on the precision of the simulations.

4.1. Defining the product $\otimes_F$

Consider three nodes A, B and C, such that A is the cause of B and B is the cause of C (Fig. 5). Arc AB carries the transfer function f_1 and arc BC carries f_2. Node B must be eliminated to reduce this path to a single arc.

It is clear that, for arcs in series, the total delay is the sum of the delays of each arc, and the gain is the product of the gains of each arc. It is then sufficient to determine the corresponding time constant and the most suitable order.

Assume two Strejc functions (Eq. (25)): f_1 and f_2; an approximation f (also a Strejc function) must be determined.

$$f_1 = \frac{g_1 \bullet e^{-d_1 \bullet s}}{(1 + T_1 \bullet s)^{n_1}} \quad f_2 = \frac{g_2 \bullet e^{-d_2 \bullet s}}{(1 + T_2 \bullet s)^{n_2}}$$
$$f = \frac{g \bullet e^{-d \bullet s}}{(1 + T \bullet s)^n} \tag{28}$$

The gain $g = g_1 \cdot g_2$ and the delay $d = d_1 + d_2$ are known a priori. In order to find the order and time constant, the response of the product $f_1 \cdot f_2$ and that of the approximation f to a step input will be analyzed, assuming zero delay and unit gain.

Consider y_{f1f2} the step response of the product $f_1 \cdot f_2$, where $n_1 = n_2 = 1$, and y_n the response of f.

$$y_{f_1 \cdot f_2}(t) = 1 - \frac{T_1}{T_1 - T_2} \cdot e^{-\frac{t}{T_1}} - \frac{T_2}{T_2 - T_1} \cdot e^{-\frac{t}{T_2}} \tag{29}$$

$$y_n(t) = 1 - \sum_{i=0}^{n-1} \frac{t^i}{i! \cdot T^i} \cdot e^{-\frac{t}{T}} \tag{30}$$

The integral error e (Eq. (31)) between the responses (Eqs. (29) and (30)) is used to minimize their relative distance.

$$e = \int_0^{+\infty} \left(\frac{T_1}{T_1 - T_2} \cdot e^{-\frac{t}{T_1}} + \frac{T_2}{T_2 - T_1} \cdot e^{-\frac{t}{T_2}} - \sum_{i=0}^{n-1} \frac{t^i}{i! \cdot T^i} \cdot e^{-\frac{t}{T}} \right) \mathrm{d}t \quad n \in \{1,2\} \tag{31}$$

Note that:

$$\int_0^{+\infty} \sum_{i=0}^{n-1} \frac{t^i}{i! \cdot T^i} \cdot e^{-\frac{t}{T}} \mathrm{d}t = n \cdot T \tag{32}$$

The integral error is therefore:

$$e = \frac{T_1^2}{T_1 - T_2} + \frac{T_2^2}{T_2 - T_1} - n \cdot T = T_1 + T_2 - n \cdot T \tag{33}$$

(Eq. (33)) shows that the integral error is zero if:

$$T = \frac{T_1 + T_2}{n} \tag{34}$$

By similarly calculating the integral errors obtained with combinations of functions of various orders [35], it can be shown that the integral error is zero if the approximation time constant is equal to the mean of the time constants of the combined functions weighted by their order (Eq. (35)).

$$T = \frac{n_1 \cdot T_1 + n_2 \cdot T_2}{n} \tag{35}$$

The optimum order n of the approximation function must still be determined. Based on an analysis of various results obtained by combining systems with time constants of different orders of magnitude, the following relations may be proposed:

$$\begin{cases} n_1 \cdot T_1 \gg n_2 \cdot T_2 \;\Rightarrow\; n = n_1 \\ n_1 \cdot T_1 \ll n_2 \cdot T_2 \;\Rightarrow\; n = n_2 \\ n_1 \cdot T_1 \approx n_2 \cdot T_2 \;\Rightarrow\; n = n_1 + n_2 \end{cases} \tag{36}$$

The approximations are summarized in Table 1. In this table, f_1 corresponds to the columns and f_2 to the rows. An element i,j represents the approximation of $f_1 f_2$. For instance, the element (1,2) refers to (35), (36). When at least one of the functions f_1, f_2 is integral (last row or last column) or derivative (first row or first column), the product is integral or derivative unless the functions are of opposite type (elements (1,3) and (3,1)). In Table 1, it may be noted that the product of two differentiators yields a negligible output (element (1,1)). This approximation is perfectly plausible, considering the signals actually recorded for industrial processes (generally step or ramp signals). Moreover, it appears unnecessary in practice to use values of n greater than 4 for Eq. (25) or greater than 2 for Eq. (26).

4.2. Defining the sum $\oplus_F$

A causal graph gives the cause-effect relations between two variables: it is therefore impossible for two parallel arcs

Table 1
Rules for approximating the product of paths

Product $g = g_1 \cdot g_2 \quad r = r_1 + r_2$	$\frac{g_1 \cdot s \cdot e^{-r_1 \cdot s}}{(1 + T_1 \cdot s)^{n_1}} \quad n_1 \in \{0,1,2\}$	$\frac{g_1 \cdot e^{-r_1 \cdot s}}{(1 + T_1 \cdot s)^{n_1}} \quad n_1 \in \{0,1,2,3,4\}$	$\frac{g_1 \cdot e^{-r_1 \cdot s}}{s}$
$\frac{g_2 \cdot s \cdot e^{-r_2 \cdot s}}{(1 + T_2 \cdot s)^{n_2}} \quad n_2 \in \{0,1,2\}$	$\varnothing$	$\frac{g \cdot s \cdot e^{-r \cdot s}}{(1 + T \cdot s)^n}$ $\begin{cases} n_1 \cdot T_1 >> n_2 \cdot T_2 \;\Rightarrow\; n = n_1 \\ n_1 \cdot T_1 << n_2 \cdot T_2 \;\Rightarrow\; n = n_2 \\ n_1 \cdot T_1 \approx n_2 \cdot T_2 \;\Rightarrow\; n = n_1 + n_2 \end{cases}$ $\begin{cases} n = 0 \;\Rightarrow\; T = 0 \\ n \neq 0 \;\Rightarrow\; T = \frac{n_1 \cdot T_1 + n_2 \cdot T_2}{n} \end{cases}$	$\frac{g \cdot e^{-r \cdot s}}{(1 + T_2 \cdot s)^{n_2}}$
$\frac{g_2 \cdot e^{-r_2 \cdot s}}{(1 + T_2 \cdot s)^{n_2}} \quad n_2 \in \{0,1,2,3,4\}$	$\frac{g \cdot s \cdot e^{-r \cdot s}}{(1 + T \cdot s)^n}$ $\begin{cases} n_1 \cdot T_1 >> n_2 \cdot T_2 \;\Rightarrow\; n = n_1 \\ n_1 \cdot T_1 << n_2 \cdot T_2 \;\Rightarrow\; n = n_2 \\ n_1 \cdot T_1 \approx n_2 \cdot T_2 \;\Rightarrow\; n = n_1 + n_2 \end{cases}$ $\begin{cases} n = 0 \;\Rightarrow\; T = 0 \\ n \neq 0 \;\Rightarrow\; T = \frac{n_1 \cdot T_1 + n_2 \cdot T_2}{n} \end{cases}$	$\frac{g \cdot e^{-r \cdot s}}{(1 + T \cdot s)^n}$ $\begin{cases} n_1 \cdot T_1 >> n_2 \cdot T_2 \;\Rightarrow\; n = n_1 \\ n_1 \cdot T_1 << n_2 \cdot T_2 \;\Rightarrow\; n = n_2 \\ n_1 \cdot T_1 \approx n_2 \cdot T_2 \;\Rightarrow\; n = n_1 + n_2 \end{cases}$ $\begin{cases} n = 0 \;\Rightarrow\; T = 0 \\ n \neq 0 \;\Rightarrow\; T = \frac{n_1 \cdot T_1 + n_2 \cdot T_2}{n} \end{cases}$	$\frac{g \cdot e^{-r \cdot s}}{s}$
$\frac{g_2 \cdot e^{-r_2 \cdot s}}{s}$	$\frac{g \cdot e^{-r \cdot s}}{(1 + T_1 \cdot s)^{n_1}}$	$\frac{g \cdot e^{-r \cdot s}}{s}$	$\frac{g \cdot e^{-r \cdot s}}{s}$

Table 2
Rules for approximating the sum of paths with different delays

Sum $r_1 \neq r_2$		$\frac{g_1 \cdot s \cdot e^{-r_1 \cdot s}}{(1+T_1 \cdot s)^{n_1}}$	$n_1 \in \{0,1,2\}$	$\frac{g_1 \cdot e^{-r_1 \cdot s}}{(1+T_1 \cdot s)^{n_1}}$	$n_1 \in \{0,1,2,3,4\}$	$\frac{g_1 \cdot e^{-r_1 \cdot s}}{s}$
$\frac{g_2 \cdot s \cdot e^{-r_2 \cdot s}}{(1+T_2 \cdot s)^{n_2}}$	$n_2 \in \{0,1,2\}$			$\frac{g_1 \cdot e^{-r_1 \cdot s}}{(1+T_1 \cdot s)^{n_1}}$		$\frac{g_1 \cdot e^{-r_1 \cdot s}}{s}$
$\frac{g_2 \cdot e^{-r_2 \cdot s}}{(1+T_2 \cdot s)^{n_2}}$	$n_2 \in \{0,1,2,3,4\}$	$\frac{g_2 \cdot e^{-r_2 \cdot s}}{(1+T_2 \cdot s)^{n_2}}$				$\frac{g_1 \cdot e^{-r_1 \cdot s}}{s}$
$\frac{g_2 \cdot e^{-r_2 \cdot s}}{s}$		$\frac{g_2 \cdot e^{-r_2 \cdot s}}{s}$		$\frac{g_2 \cdot e^{-r_2 \cdot s}}{s}$		

to originate from the same variable and lead to the same variable. It is possible, however, for two parallel paths to meet. Seeking an approximation of the combination of two parallel paths corresponds to the approximation of a sum of two transfer functions.

It is readily observed that two paths with different delays cannot be combined in a simple manner: if a system gives two responses with a time lag to the same variable, it cannot normally be considered as a single relation. When two different delays are detected on the paths to be combined, either one of the paths must be ignored in favor of the other (any Strejc transfer or differentiator can be ignored in favor of an integrator, and any differentiator can be ignored in favor of a Strejc transfer or integrator) or the graph structure must be revised to prevent the combination of these two paths. As in the case of the transfer product, the rule for approximating the sum of paths with different delays are indicated in Table 2.

Now consider two paths with identical delays. The case of parallel paths is the same as the sum of two transfer functions: the only result that can be considered immediate is that the overall gain is equal to the sum of the gains ($g = g_1 + g_2$).

The integrator is again considered dominant; differentiators and Strejc functions can again be disregarded when one is present. Similarly, a differentiator is systematically ignored in the presence of a Strejc function. Only the sums of transfer functions of the same type will be examined.

Consider first the sum of two Strejc functions (Eq. (37)).

$$f_1 = \frac{g_1 \cdot e^{-d \cdot s}}{(1+T_1 \cdot s)^{n_1}} \quad f_2 = \frac{g_2 \cdot e^{-d \cdot s}}{(1+T_2 \cdot s)^{n_2}}$$
$$f = \frac{g \cdot e^{-d \cdot s}}{(1+T \cdot s)^{n}} \tag{37}$$

The sum of the two first orders ($n_1 = n_2 = 1$) has three different shapes (Fig. 7), in addition to the trivial case of the difference of two strictly identical transfers. The response is similar to a first-order one if the gains are both of the same sign. It is similar to a second-order differentiator if the gains are opposite. The characteristic 'hump' of a non-negligible zero appears only if the gains are of different signs and different absolute values (noted as 'different signs' in Fig. 7). This phenomenon affects only part of the transient, and was therefore disregarded.

The results are similar for higher-order transfers.

As for the product, the response of two Strejc transfers is approximated by one Strejc transfer by eliminating the integral error (Eq. (38)).

$$\int_0^{+\infty} \left(g_1 \sum_{i=0}^{n_1-1} \frac{t^i}{i! \cdot T_1^i} \cdot e^{-\frac{t}{T_1}} + g_2 \sum_{j=0}^{n_2-1} \frac{t^j}{j! \cdot T_2^j} \cdot e^{-\frac{t}{T_2}} - (g_1+g_2) \sum_{k=0}^{n-1} \frac{t^k}{k! \cdot T^k} \cdot e^{-\frac{t}{T}} \right) \mathrm{d}t = n_1 \cdot g_1 \cdot T_1 + n_2 \cdot g_2 \cdot T_2 - n(g_1+g_2)T \tag{38}$$

With a non-zero overall gain the sum of Strejc functions is approximated by a Strejc function of the lowest order (Fig. 7). This method retains the simplest dynamics. The equivalent time constant should be the mean time constant weighted by the orders and gains, provided it is positive. Hence the following two cases:

$$T = \frac{\mathrm{Max}(n_1 \cdot T_1, n_2 \cdot T_2)}{n}$$
$$g = g_2 + g_1 \; n = \mathrm{Min}(n_1, n_2) \;\; g_2 \cdot g_1 < 0 \tag{39}$$

$$T = \frac{n_1 \cdot g_1 \cdot T_1 + n_2 \cdot g_2 \cdot T_2}{n \cdot g}$$
$$g = g_2 + g_1 \; n = \mathrm{Min}(n_1, n_2) \;\; g_2 \cdot g_1 > 0 \tag{40}$$

When the overall gain ($g_1 + g_2$) is zero, the approximation requires a second-order differentiator to retain the shape

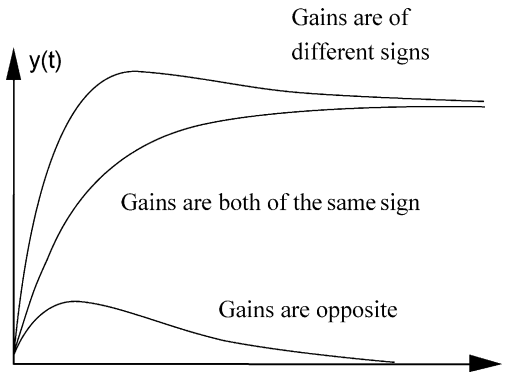


Fig. 7. The sum of the two first orders $n_1 = n_2 = 1$ has three different shapes.

Table 3
Rules for approximating the sum of paths with identical delays

Sum	$\frac{g_1 \bullet s \bullet e^{-r_1 \bullet s}}{(1+T_1 \bullet s)^{n_1}}$			$\frac{g_1 \bullet e^{-r_1 \bullet s}}{(1+T_1 \bullet s)^{n_1}}$			$\frac{g_1 \bullet e^{-r_1 \bullet s}}{s}$
$r = r_1 = r_2$	$n_1 \in \{0,1,2\}$			$n_1 \in \{0,1,2,3\}$			
	$g_1 \bullet g_2 > 0$	$g_1 + g_2 = 0$	$g_1 \bullet g_2 < 0$		$g_1 + g_2 = 0$	$g_1 \bullet g_2 < 0$	
$\frac{g_2 \bullet s \bullet e^{-r_2 \bullet s}}{(1+T_2 \bullet s)^{n_2}}$	$\frac{g \bullet s \bullet e^{-r \bullet s}}{(1+T \bullet s)^n}$	$\varnothing$	$\frac{g \bullet s \bullet e^{-r \bullet s}}{(1+T \bullet s)^n}$	$\frac{g_1 \bullet e^{-r \bullet s}}{(1+T_1 \bullet s)^{n_1}}$	$\frac{g \bullet s \bullet e^{-r \bullet s}}{(1+T \bullet s)^2}$	$\frac{g \bullet e^{-r \bullet s}}{(1+T \bullet s)^n}$	$\frac{g_1 \bullet e^{-r \bullet s}}{s}$
$n_2 \in \{0,1,2\}$	$g = g_1 + g_2$		$g = g_1 + g_2$		$g = g_1 \bullet n_2 \bullet T_2 + g_2 \bullet n_1 \bullet T_1$	$g = g_1 + g_2$	
	$n = \text{Min}(n_1, n_2)$		$n = \text{Min}(n_1, n_2)$		$n = 2$	$n = \text{Min}(n_1, n_2)$	
	$T = \frac{\text{Max}(n_1 \bullet T_1 \bullet n_2 \bullet T_2)}{n}$				$T = \frac{\text{Max}(n_1 \bullet T_1, n_2 \bullet T_2)}{n}$	$T = \frac{\text{Max}(n_1 \bullet T_1, n_2 \bullet T_2)}{n}$	
	$\frac{g_1 \bullet n_1 \bullet T_1 + g_2 \bullet n_2 \bullet T_2}{n \bullet g}$			$g_1 \bullet g_2 > 0$			
$\frac{g_2 \bullet e^{-r_2 \bullet s}}{(1+T_2 \bullet s)^{n_2}}$	$\frac{g \bullet e^{-r \bullet s}}{(1+T_2 \bullet s)^{n_2}}$			$\frac{g \bullet s \bullet e^{-r \bullet s}}{(1+T \bullet s)^n}$			
$n_2 \in \{0,1,2,3\}$				$g = g_1 + g_2$			
				$n = \text{Min}(n_1, n_2)$			
				$T = \frac{g_1 \bullet n_1 \bullet T_1 + g_2 \bullet n_2 \bullet T_2}{n \bullet g}$			
$\frac{g_2 \bullet e^{-r_2 \bullet s}}{s}$	$\frac{g_2 \bullet e^{-r \bullet s}}{s}$			$\frac{g_2 \bullet e^{-r \bullet s}}{s}$			$\frac{(g_1+g_2) \bullet e^{-r \bullet s}}{s}$

recognizable in Fig. 7.

$$T = \frac{\text{Max}(n_1 \bullet T_1, n_2 \bullet T_2)}{2}$$
$$g = n_1 T_1 g_2 + n_2 T_2 g_1 \quad n = 2 \tag{41}$$

Two integrators are approximated simply by considering them as an integrator with a gain equal to the sum of the gains $g_1 + g_2$.

Two differentiators are approximated as for two Strejc functions. When the overall gain $g_1 + g_2$ is zero, the resulting behavior is negligible. When both gains are of the same sign, the mean time constant weighted by the orders and gains is used; if the gains are of opposite signs, the approximation corresponds to Eq. (39).

The results of these approximations are indicated in Table 3 for two paths with identical delays.

5. Industrial application

5.1. Pulsed columns example

The hierarchical decomposition method was tested on a nuclear fuel reprocessing plant [3]. The process unit under consideration is a pulsed column facility, comprising two head-to-tail pulsed columns, coupled by means of a transfer system and their feed systems. A pulsed column is a liquid-liquid extractor used to separate uranium and plutonium, initially in an acid solution, from fission products, which are nuclear wastes. The two main product streams contain an organic phase and an aqueous phase (Fig. 8). Transfer systems between the two columns may be ignored in a global graph, or considered in detail in a low-hierarchy graph. Thus it is easy to know the variables attached to each function.

Let's consider the extraction column. The in-flowing aqueous stream includes two flows (Q0500 and QG100); it is mixed with the organic solution (Q1010) in the shaft of the column. The out-flowing acid solution (QE120) contains

Fig. 8. Schematic view of the plant.

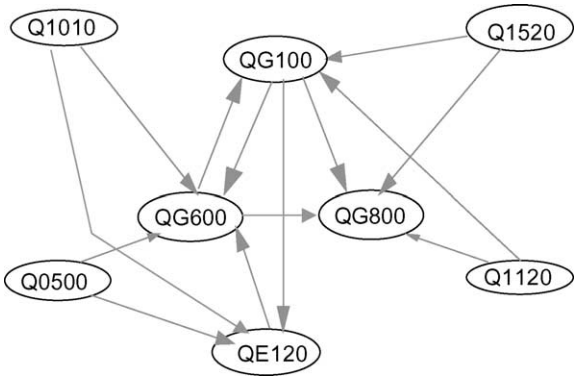


Fig. 9. Highest level causal graph of two coupled columns.

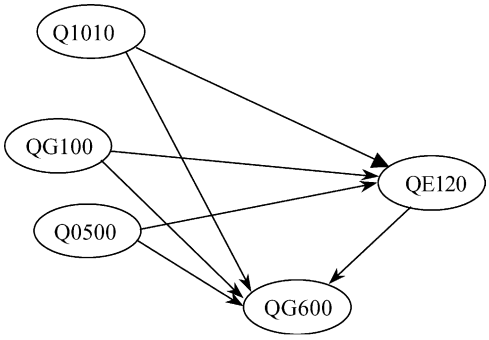


Fig. 10. The mass balance graph for a single column.

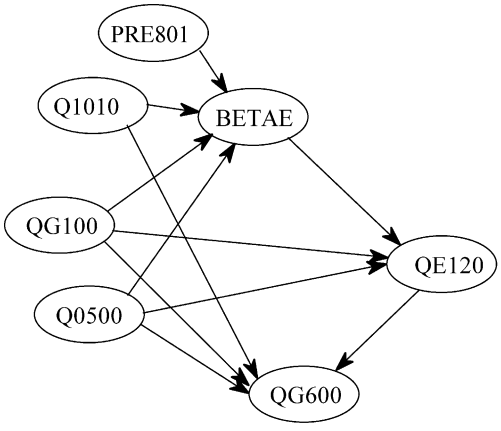
only the fission products, which are subsequently confined in a glass matrix. The out-flowing organic solution (QG600) is removed by overflow (Fig. 8).

The extraction column is followed by a scrubbing column designed to enhance the decontamination of fission products from the solvent (aqueous phase: inlet Q1120 and Q1520, outlet QG100; organic phase: inlet QPAR; outlet QG800). Traces of fission products carried out from the extraction column (QG600) are rinsed from the organic phase containing the uranium and the plutonium. The out-flowing aqueous solution QG100 is fed back to the extraction column in order to recover the aqueous phase containing the fission products through a single outflow (QE120).

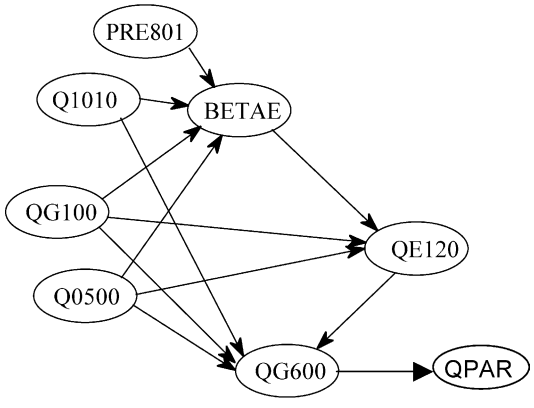
The highest-level graph represents the columns hydraulic balances, using variables Q0500, QG100, Q1010, QG600, QE120, Q1120, Q1520, QG800, among which the causal relations must be determined. The global balance of the aqueous phase and the solvent, the balances between the solvent inflow and outflow rates and between the aqueous phase outflow and overflow rates result in a graph with eight nodes and fourteen arcs, represented in Fig. 9. It represents the hydraulic balance at the level of the global process. The set-point changes of the in-flowing streams are the most frequent ones in the control of the facility; other set-point changes are to be considered exceptional. As a consequence, the hydraulic balances constitute the best information about the facility global state. When this balance is verified, the operators consider the process to be in normal operation. The availability of the plant is not questionable. They are not interested in monitoring other variables than those represented on this graph. Only changes in other control inputs that they have to make unusually or an alarm on any monitored variable will draw their attention to a more detailed view of the installation.

The unit functional decomposition into the extraction and the washing columns is natural. In the following analysis, only the example of the extraction column is considered, in order to simplify the explanations. Thus, the highest causal graph in the hierarchy is the one in Fig. 10.

Extraction requires suitable mixing of the aqueous and organic phases to maximize the contact surface area between them and thus optimize the chemical exchanges. This exchange efficiency is estimated by the column retention BETAE, a non-measurable variable that is a function of the rate of aqueous phase in the continuous organic phase. The mixture is subject to a periodic pulsation pressure (PRE801) to slow the descent of the heavy aqueous phase and form an emulsion with the organic phase. The second level of the hierarchy includes the retention and the pressure. All the inflows influence



11-a



11-b

Fig. 11. The second and third level of decomposition for a single column.

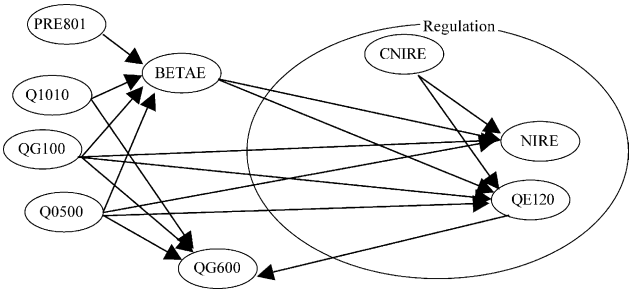


Fig. 12. The fourth level graph with inter-phase level regulation.

the retention, which also depends on the pressure. The second-level graph for the extraction function thus contains seven nodes and 11 arcs (Fig. 11-a). A new path has been added from PRE801 to BETAE that allows interpretation of the flows transitory behavior.

In the upper graph, QG600 was considered to be the organic inflow of the washing column. In a more detailed representation, the organic outflow of the extraction column and the organic inflow of the washing column must be distinguished. The buffer tank and the transfer device between the two columns are thus taken into account and the arc (QG600, QPAR) is added to the previous graph (Fig. 11b). The same work can be done for all the feeding systems of the columns but will not be detailed here.

The inter-phase represents the physical boundary between the two phases in the lower settling tank. It is regulated by the aqueous outlet flow rate (QE120); the inter-phase level is measured (NIRE).

The fourth hierarchical level includes the column inter-phase level (NIRE) regulation. CNIRE is the setpoint. The column inflow rates disturb NIRE: Q0500 and Q1010 are to be considered as measurable disturbances of this control loop.

The regulation decomposition proposed in Section 3 is applied to NIRE regulation and results in Fig. 12. Via the regulation system, paths replace the arcs used in the first and second levels. The same kind of decomposition applies to Q1010, Q0500, and PRE801 which are regulated variables. A low-level graph contains the details of all these regulations.

BETAE can be split into three variables, expressing the retention rate at several heights in the column.

The most detailed graph, including all the transfer systems serving as buffer between the columns, includes 55 nodes (Fig. 13) [36].

In Fig. 14a, the set point CQ1010 of the organic inflow of the first column is varied following a ramp signal and takes its steady value at time 2200. During the transient behavior, the more detailed the variables' evolution, the easier the operators' analysis. Thus, the operators first wait for the column retentions BETAE and BETAL be stabilized (BETAL is the equivalent of BETAE for the second column) (Fig. 14b). During this transient, the detailed graph (Fig. 13) is used both for variables computation and Man–Machine Interface. After that, the level of detail required by the operators for their supervision tasks decreases: an abstract graph is then sufficient as soon as the steady state is reached. The evolution of the organic

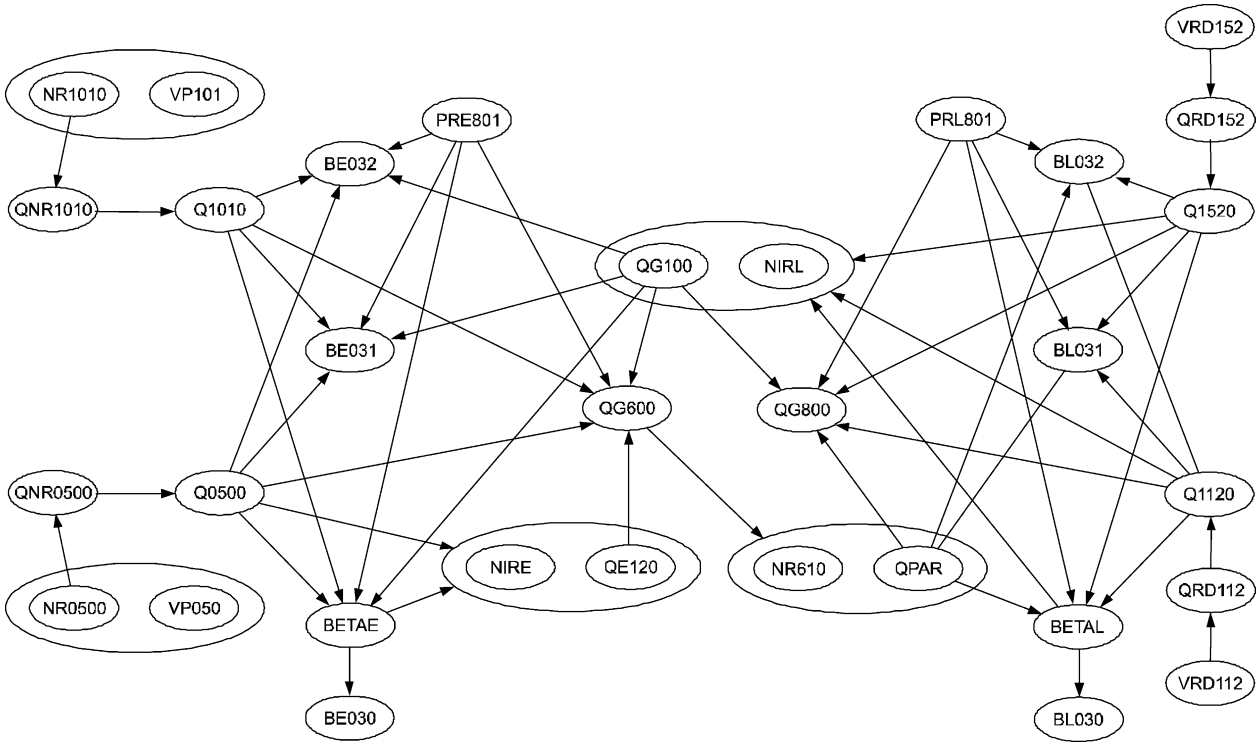


Fig. 13. Complete causal graph of the two columns.

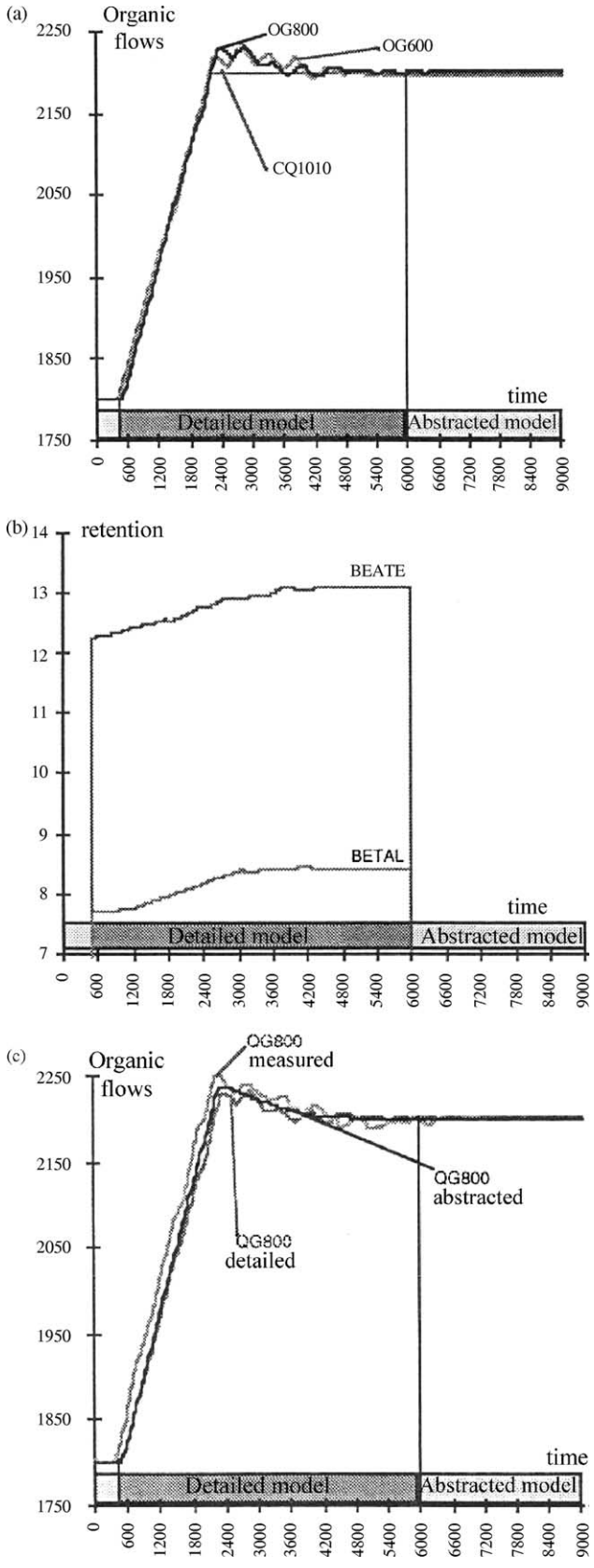


Fig. 14. (a) Evolutions of the organic flows with the detailed and abstracted model. (b) Evolutions of the retention variables with the detailed model. (c) Outflow QG800: measured, computed with the detailed and abstracted model.

inflows and outflows with the detailed model are presented in Fig. 14a until the steady state at time 6000; then these evolutions result from the abstracted model.

To assess the difference between the evolution obtained with the detailed model and the simplified model, Fig. 14c shows three plots for the organic outflow QG800; the pattern obtained with the simplified model is superimposed on the one obtained with the detailed model between times 500 and 6000. Fig. 14c also shows the measured evolution of QG800 under the same conditions. The approximate transfer functions of the simplified graph were obtained as described in Section 4.

5.2. Industrial usage

In 1987, the French Atomic Commission (CEA) initiated an R & D program, DIAPASON, in the area of continuous process supervision. It was intended to promote the use of model-based diagnosis in nuclear plants. The size, the degree of automation and the complexity of the facilities in a spent nuclear fuel reprocessing plant make it a focus of attention for supervision applications. The dynamical causal graph was the support of the simulation and explanation functionalities of the prototype DIAPASON [17,37]. Based on the knowledge of the dynamic causal dependencies among the graph variables, algorithms were implemented to identify the first deviation capable of explaining all the observed alarms. This led to the development of a comprehensive fault detection, isolation and alarm filtering system integrated in DIAPASON. The prototype was tested on a very accurate numerical simulator of the head-to-tail pulsed column process [13,19].

A second program, SUP-XX, demonstrated in 1998 the feasibility of many of advanced operator support features in an industrial context [3].

A third prototype, SALOMON II, supported by the *Direction de la Recherche et de la Technologie de la Division Générale de l'Armement* (DRET), was tested to evaluate the benefits of causal representations with experienced and novice operators. The offered functionalities were very appreciated by both groups: the conclusion of the evaluation experiments was that the causal graph was a powerful tool for those problems that are not easily managed by novice operators, such as faults in regulation or aqueous-organic phases inversion [36].

Despite these encouraging results, experimentations have shown that the diversity of the functions that must be executed by process control operators leads them to construct representations involving several levels of abstraction concerning process operation in order to obtain the relevant level of detail for each task. It was thus necessary to implement in the interface several causal graphs including various levels of detail. The operators could choose on-line the graph they found the most relevant, but in an a priori defined graph hierarchy [38]. Computations were driven at the most detailed level.

The on-line construction of a causal graphs hierarchy presented in this paper has not yet been implemented in the industrial prototype but simply experimented in laboratory. Nevertheless, the proposed approach constitutes a first step towards human-computer adaptive interface.

6. Conclusion

The work discussed in this paper presents an approach to the problem of hierarchical modeling of complex processes for supervision purposes. The model is designed for operator assistance and is usable in a cooperative approach. Causality and time-driven reasoning are fundamental aspects of operator reasoning. In this paper, a causal dynamic model of the process to be supervised supports operator reasoning. The model is represented by a directed graph reflecting the causality of the physical phenomena, with time-semantic parameters attached to the arcs.

Considering the variety of operator tasks, however, and the knowledge required for a satisfactory understanding of process operation, it is not reasonable to use a single graph to represent an entire plant. An original method for construction and use of a directed graph hierarchy is proposed in this paper. Abstract models appear at the top of the hierarchy and the most detailed ones at the bottom. This multi-level representation is well adapted to human decision making because the varying levels of detail are adapted to the various supervision tasks.

During process operation, the operators are provided with a context-dependent graph. Such graphs differ by the number of variables included and are constructed on-line at the initiative of the operators who select the relevant variables they want to monitor.

Path algebra provides an elegant tool for representing a graph by a matrix, counting the paths, listing them by name, or identifying loops. A procedure is proposed to eliminate each of the non-relevant nodes from the graph in an iterative manner to obtain the required level of abstraction. Each step is reduced to matrix products, which successively eliminate each non-relevant variable while conserving its influence on the relevant variables. However, paths comprising loops cannot all be eliminated; at least two variables must be retained per loop, for example those with the largest number of external links. This restriction is not applicable to control loops, which can easily be represented by a causal structure without loops.

The matrix coefficients are the information assigned to the arcs in the graph. The matrix sums and products must therefore be redefined as the sums and products of the information carried by the arcs. Simple indicators (e.g. signs or orders of magnitude of influence) are easily combined. In this work, the arcs carry time-related information in the form of transfer functions. The transfer functions representing the process have been included in a reduced library with few parameters. This allows very rapid simulation. The transfer functions of the detailed model must then be merged on-line to obtain the parameters of the higher level models. In order to obtain the matrix product required by path algebra, the sums and products of the transfer functions were redefined to provide approximations conserving the delays, static gains and response times. Not all combinations can be reduced to simple behavior patterns and some constraints must be verified by the operator-designated graph.

Some points are still to be studied before the proposed methodology becomes a complete industrial tool.

The variables selection by the operators assumes they have considerable decision making autonomy and know what should be displayed for assistance purposes. This is a valid assumption except under stress situations. Thus verification tools should be provided, imposing in some situations the view of the part of the process that is suspected of being faulty.

In many industrial situations, processes can work with different configurations. These correspond for instance to different operating modes (start up, shut down, failure rejection, etc.). Configuration changes are managed by the supervision system, which could at the same time propose the corresponding interface to the operators. In this case, the change is not human driven but automatic. The operator is always presented the graph corresponding to the current configuration and can move in a hierarchy corresponding to this configuration.

The results discussed in this paper constitute a preliminary approach to the problem of intelligent interface design, which is the next step of the work: the supervision interface can reproduce the selected graph that in its turn reflects the actual process state; the computations are adapted to this interface; selecting the variables of interest can be envisaged with simple mouse actions from the operator.

This type of interface can be envisaged not only for supervision but also for educational purposes of novice operators because causality is a good explanation tool of what they are observing. Having to choose the right level in a graph hierarchy makes them think about the level of detail necessary to understand a developing supervision problem. The necessity of configuring dynamically their interface keeps operators active during their tasks, which is another good point for intelligent interfaces.

References

[1] Rasmussen J. Information processing and human–machine interaction. Amsterdam: North Holland; 1986.

[2] Leitch R, Stefanini A. Task dependent tools for intelligent automation. Artific Intell Eng 1989;4(3):126–43.

[3] Montmain J. Supervision applied to nuclear fuel reprocessing. AI Commun, Eur J AI 2000;13(2):61–81.
[4] Rouse W. Models of human problem solving: detection, diagnosis, and compensation for system failures. Automatica 1983;19(6): 613–25.
[5] Hoc J-M. Towards a cognitive approach to human–machine cooperation in dynamic situations. Int J Human–Comput Stud 2001; 54(4):509–40.
[6] Lind M. Status and challenges of intelligent plant control. Annu Rev Control 1996;20:23–41.
[7] Millot P, Pacaux-Lemoine AM. An attempt for generic concepts toward Human Machine cooperation. IEEE SMC Conference San Diego, USA; 1998.
[8] Johanssen G, Levis A, Stassen H. Theoretical problems in man–machine systems and their experimental validation. Automatica 1994; 30(2):217–31.
[9] Struss P. AI methods for model-based diagnosis. 12th International Workshop on Principles of Diagnosis DX01-Bridge Workshop 2001, Via Lattea, Italy; 2001.
[10] Isermann R. Supervision, fault-detection and fault-diagnosis methods—an introduction. Control Eng Practice 1997;5(5):639–52.
[11] Cacciabue P, Decortis F, Drodzdowicz B, Masson H, Nordvik JP. Cosimo: a cognitive simulation model of human decision making and behavior in accident management of complex plants. IEEE Trans Syst, Man Cybern 1992;22(5):1058–74.
[12] Evsukoff A, Gentil S, Montmain J. Fuzzy reasoning in co-operative supervision systems. Control Eng Practice 2000;8:389–407.
[13] Montmain J, Gentil S. Dynamical causal model diagnostic reasoning for on-line technical process supervision. Automatica 2000;36: 1137–52.
[14] Rasmussen J. Diagnostic reasoning in action. IEEE Trans Syst Man Cybern 1993;23(4):981–91.
[15] Iwasaki Y, Simon HA. Causality in Device Behaviour. Artif Intell 1986;29(1):3–33.
[16] de Kleer J, Brown JS. A Qualitative Physics Based on Confluences. Artif Intell 1984;24:7–83.
[17] Leyval L, Gentil S, Feray-Beaumont. Model based causal reasoning for process supervision. Automatica 1994;30(8):1295–306.
[18] Trave-Massuyes L, Milne R. Diagnosis of dynamic systems based on explicit and implicit behavioral models: an application to gas turbines in Esprit Project TIGER. Appl Artif Intell J 1996;10(3): 257–77.
[19] Montmain J, Leyval L, Gentil S. Qualitative analysis for decision making in supervision of industrial continuous processes. Math Comput Simul 1994;36:149–63.
[20] Console L, Torasso P. An approach to the compilation of operational knowledge from causal models. IEEE Trans Syst, Man Cybern 1992; 22(4):772–89.
[21] Darwiche A. Model-based diagnosis using structured system descriptions. J AI Res 1998;8:165–222.
[22] Gentil S, Montmain J, Combastel C. Causal model based diagnosis: comparison with other approaches. IEEE Trans Syst, Man Cybern—Part B 2004;34(5):2207–21.
[23] Rasmussen J. The role of hierarchical knowledge representation in decision making and system management. IEEE Trans Syst, Man Cybern 1985;15(2):234–43.
[24] Iwasaki Y. Reasoning with multiple abstraction models. Fourth International Workshop on Qualitative Physics, Lugano, Switzerland; 1990.
[25] Iri M, Aoki K, O'Shima E, Matsuyama H. An algorithm for diagnosis of system failures in the chemical process. Comput Chem Eng 1979;3: 489–93.
[26] Shiozaki J, Matsuyama H, Tano K, O'Shima E. Fault diagnosis of chemical processes by the use of signed directed graphs. Int Chem Eng 1985;25(4):651–9.
[27] Kramer MA, Palowitch B. A rule based approach to fault diagnosis using the signed directed graph. Am Inst Chem Engnr J 1987;33(7): 1067–78.
[28] Yu C, Lee C. Fault Diagnosis Based on Qualitative/Quantitative Process Knowledge. AIChE J 1991;37(4):617–27.
[29] Montmain J, Gentil S. Operation support for alarm filtering. IEEE Conference Computational Engineering in System Applications, Lille, France; 1996.
[30] Mosterman PJ, Biswas G. Diagnosis of continuous valued systems in transient operating regions. IEEE SMC—Part A: Syst Humans 1999; 29(6):554–65.
[31] Gondran M, Minoux M. Graphs and algorithms. New York: Wiley; 1984.
[32] Dziopa P, Leyval L, Gentil S. Qualitative reasoning for supervision. International Joint Conference on Artificial Intelligence, Qualitative Reasoning Workshop, Chambéry, France; 1993.
[33] Ljung L. System identification: theory for the user. Englewood Cliffs, NJ: Prentice-Hall; 1987.
[34] Strejc V. Détermination approchée des caractéristiques de régulation d'un processus à réponse apériodique. Automatisme 1960;5(3): 109–11.
[35] Gentil S, Dziopa P, Montmain J. Hierarchical causal model for complex plant supervision. Third IMACS—IEEE international multiconference on circuits, systems, communications and computers, Athens, Greece; 1999.
[36] Evsukoff A, Montmain J, Gentil S. Causal model based supervising and training. IFAC workshop on supervision of chemical industries, Lyon, France; 1998.
[37] Penalva JM, Coudouneau L, Leyval L, Montmain J. Diapason: a supervision support system. IEEE Expert Intell Syst Appl 1993;8(5): 57–65.
[38] Lambert M, Riera B, Martel G. Realization and evaluation of a new kind of supervisory system. Proceedings of the 7th IFAC/IFIP/IFORS/EA Symposium on Analysis, Design and Evaluation of Man-Machine Systems, Kyoto, Japan; 1998.