Locating Sensors in Complex Chemical Plants Based on Fault Diagnostic Observability Criteria

Rao Raghuraj, Mani Bhushan, and Raghunathan Rengaswamy
Dept. of Chemical Engineering, Indian Institute of Technology, Bombay Powai, Mumbai-400 076, India

Fault diagnosis is an important task for the safe and optimal operation of chemical processes. Hence, this area has attracted considerable attention from researchers in the past few years. A variety of approaches have been proposed for solving this problem. All approaches for fault detection and diagnosis in some sense involve the comparison of the observed behavior of the process to a reference model. The process behavior is inferred using sensors measuring the important variables in the process. Hence, the efficiency of the diagnostic approach depends critically on the location of sensors monitoring the process variables. The emphasis of most of the work on fault diagnosis has been more on procedures to perform diagnosis given a set of sensors and less on the actual location of sensors for efficient identification of faults. A digraph-based approach is proposed for the problem of sensor location for identification of faults. Various graph algorithms that use the developed digraph in deciding the location of sensors based on the concepts of observability and resolution are discussed. Simple examples are provided to explain the algorithms, and a complex FCCU case study is also discussed to underscore the utility of the algorithm for large flow sheets. The significance and scope of the proposed algorithms are highlighted.

Introduction

The process disturbances or faults, if undetected, have a serious impact on process economy, product quality, safety, productivity, and pollution level. In order to detect, diagnose, and correct these abnormal process behaviors, efficient and advanced automated diagnostic systems are of great importance to modern complex chemical industries. Considerable research has gone into the development of such diagnostic systems. Expert systems, neural networks, rule-based techniques, and signed directed graphs are some of the diagnostic approaches discussed in the literature.

These techniques basically involve the following steps.

1. *Fault detection:* Observing the fault symptoms in measured values and detecting that a fault has occurred.
2. *Fault identification:*
   (a) *Classification:* Determining the type of the fault.
   (b) *Estimation:* Determining the extent of the fault.
   (c) *Diagnosis:* Determining the root cause of the fault.
3. *Correction:* Correcting the system by repairing or replacing.

Whenever a process encounters a fault, the effect of this fault is propagated to all or some of the process variables. The main objective of the fault-diagnosis step is to observe these fault symptoms and determine the root cause for the observed behavior. The fault-detection step involves comparing the observed behavior of the process to a reference model. This observed fault-symptom pattern forms the basis for the fault-identification step. Thus the efficiency of the diagnostic system depends critically upon the location of the sensors monitoring important process variables. With hundreds of process variables available for measurement in any chemical plant, selection of crucial and optimum sensor positions poses a unique problem. Hence there is a need for an automated procedure to design a cost-optimum, foolproof, and highly reliable fault-monitoring system for the safe operation of a typical industrial process.

Knowledge of fault propagation within the system is necessary for developing a solution to the preceding problem. Dif-

Correspondence concerning this article should be addressed to R. Rengaswamy.

ferent approaches, such as fault trees and digraphs, are available in the literature to obtain fault propagation behavior in the presence of a fault. A fault tree is a logic tree that propagates primary events or faults to a top-level event or a hazard. Hence, such a tree would give the propagation of the effect of a fault to various process variables. Signed digraph (SDG) is another technique that is concerned with the cause–effect (CE) analysis of the system (Iri et al., 1979; O'Shima et al., 1985). Iri et al. (1979) were the first to introduce an algorithm for such a CE diagnosis of system failures, based on signed directed graphs (SDG). Kramer and Palowitch (1987) developed a rule-based approach for identifying the possible causes of process disturbances using the SDG representation. Chang and Yu (1990) also gave a systematic design procedure for constructing the rule-based fault-diagnostic system using the SDG. They simplified the SDG according to states, so as to overcome problems associated with spurious and erroneous interpretations in the SDG. Mohindra and Clark (1993) used the SDG to come up with a distributed fault-diagnosis methodology.

Some work has already been done on using these techniques to locate sensors. Lambert (1977) used fault-trees to analyze the location of sensors depending on the effect of basic units (fault origins) on the process variables. Even though this work was the first step toward the design of sensor locations based on a diagnostic observability criterion, it had drawbacks such as: (1) inability to handle cycles, and (2) the development of a fault tree is in itself an error-prone and time-consuming process. A quantitative approach for sensor network design based on failure probabilities was also explained by Lambert (1977). To solve the problem of observability, based on the process graph, Ali and Narasimhan have presented sensor network design strategies for linear (Ali and Narasimhan, 1993, 1995) and bilinear (Ali and Narasimhan, 1996) processes. They dealt with multicomponent mass flow processes and energy distribution networks. Using the concepts of fault observability and fault resolution, Chang et al. (1993) developed a new optimal design strategy for fault-monitoring systems. A trial-and-error algorithm was proposed that utilizes the concept of a diagnostic efficiency table. The ability to utilize quantitative failure probability data was not incorporated in their approach. There are also other methods based on linear and bilinear mathematical models for sensor location (Madron and Veverka, 1992). These methods are not reviewed here because the solution strategy proposed in this article is fundamentally different. We approach the sensor-location problem through a qualitative CE analysis of various fault scenarios.

The importance of the problem of sensor location is evident, as all the fault-diagnosis techniques depend on a given set of observed fault symptoms. The emphasis of most of the work on fault diagnosis has been more on procedures to perform diagnosis given a set of sensors and less on the actual location of sensors for efficient identification of faults. In this article, a digraph-based approach is proposed for the problem of sensor location for the identification of faults. The digraph-based approach has many advantages. It involves a simple graphical representation of the process that is easy to analyze. A few researchers have already looked at the problem of automatic development of digraphs from the process model equations (Iri et al., 1979; Mylaraswamy et al., 1994). The qualitative SDG model is sufficient to observe the effects of faults on different process variables, avoiding complex mathematical relationships between various process variables.

In our solution to the problem of sensor location, digraphs are used to infer the propagation behavior when different faults occur. The sensor location problem is posed at two levels. First, the problem of identifying a minimum number of sensor locations for observing all the faults is formulated and solved. Second, the problem of maximum resolution of faults under single-fault and multiple-fault assumptions is formulated. It is shown that the same algorithm proposed for the observability problem can be used in solving these problems also.

Qualitative Analysis of Sensor Location

Signed-directed-graph approach

The ultimate aim of this work is the design of efficient monitoring systems that will help in quick and efficient identification of faults. The directed graph (DG) that represents the CE behavior of the process is used as a basis for the sensor-location algorithm. Most of the work done on fault diagnosis uses the SDG representing the process. The only difference between DG and SDG is that signs are placed on the arcs of DG to get an SDG. However, the structure of the DG and SDG are exactly the same. Hence, in this section we will briefly review the work done on using SDG in process fault diagnosis to emphasize the importance of proper sensor location for efficient fault diagnosis. SDG is a graphical representation of the system that utilizes nodes and branches. The nodes correspond to the process state variables and malfunctions, and the branches represent the causal relationships between the process variables. A given system can be represented structurally in the form of an SDG. All possible variables and parameters of the system are clearly defined and are represented as nodes. The model equations are properly arranged, and the effects of a particular variable on other nodes are obtained. The branch sign is fixed according to the relationship between the variables represented as its end nodes. The positive and negative influences are shown by assigning + and − signs, respectively, to the branches. The states of the variables are represented by assigning nodes +, 0, or −signs for high, normal, and low states, respectively. This representation helps in defining a pattern of observed symptoms on the process SDG. A nonzero node sign signifies the presence of a failure in the process, and a set of nonzero signs in the SDG represents a pattern of fault symptoms. It is well understood that all the variables of the process in a complex chemical plant cannot be measured due to technical and economical infeasibilities. The pattern defined is therefore always partially observed and is hence called ''the partial pattern.''

This partial pattern can be used to obtain the CE graph to find the structure of fault propagation. The CE graph consists of so-called strongly connected components (Iri et al., 1979). To aid the fault-diagnosis studies, a node or a cycle with no input arcs is considered to be a maximally strongly connected component (MSCC) in the SDG. A branch is said to be consistent if its sign is equal to the product of both of the node signs it is connected with. And a nonzero node is

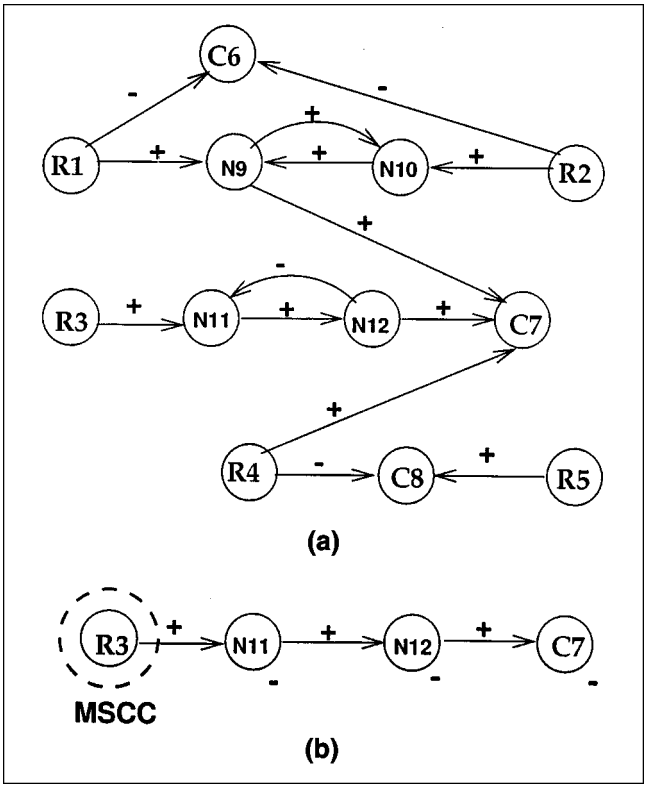

Figure 1. Explanatory digraph: (a) SDG; (b) CE graph for fault *R*3.

marked as a valid node. When a process network is decomposed into a CE graph, involving only the consistent branches and the valid nodes, the faults are assumed to originate from an element in one of the MSCCs (Iri et al., 1979). This assumption that only nodes in the maximally strongly connected components can be the root causes of faults is a trivial assumption that can be made without any loss of generality. If there is a fault node that is not a maximally strongly connected component, then a pseudonode with just one output arc can be attached to this fault node. Now in our treatment, this pseudonode thus becomes the fault node corresponding to the original fault node.

After decomposition of the process graph into a CE digraph, the CE digraph has a *root node* (an MSCC) and the corresponding reduced digraph, called a *rooted digraph.* If the single-fault assumption is valid [one fault at a time (Iri et al., 1979; Kramer and Palowitch, 1987)], then there exists a single MSCC in the rooted digraph. Figure 1 explains the formation of a rooted digraph for a particular fault in the system. Figure 1a is an SDG with different nodes representing different variables. A fault $R3$ ($R3 = -1$) is set and the symptoms are observed on different nodes. Figure 1b represents the rooted digraph for the fault $R3$ that forms an MSCC in the CE graph. Nodes $N11$, $N12$, and $C7$ are the valid nodes, with nonzero states. If fault $R3$ occurs in the process at any given time, its symptoms can be seen on the variables represented by the valid nodes. The arc from $N12$ to $N11$ is inconsistent based on the assumption that an ''effect cannot compensate for its own cause'' (Kramer and Palowitch, 1987), and has been cut in the rooted digraph. Similarly, every fault in the process has its own rooted digraph and valid nodes in the SDG, representing the process variables that are influenced by that particular fault. Thus the problem of sensor location can be viewed as a systematic procedure of identifying the root nodes corresponding to all the faults and placing the sensors on the measurable valid nodes in the CE graph. The partial pattern obtained from these sensors then helps in detecting symptoms of every fault with a given resolution.

Since SDGs have been predominantly used in fault diagnosis, we provided a brief summary of the use of SDG in that manner. In this work, however, we work only with the DG for deciding sensor locations. Since we will be dealing only with the DG, occurrence of a fault will necessarily cause the variables connected with that fault to acquire abnormal states. Therefore the arcs of a DG represent a ''will cause'' relationship, that is, an arc from node A to node B implies that A is a sufficient condition for B. This in general is not true for an SDG. As an example, the effect of a fault on a connected variable might be offset by the opposing effect of some other fault on that variable. Another example might be the case of a controlled variable that would assume a normal value at steady state, though there would be nodes with abnormal values connected to it. Therefore the arcs in the SDG represent a ''can cause'' relationship, that is, an arc from node A to node B only implies that A can cause B, and not that A will cause B. If the signs of the arcs were to be considered, then suitable modifications can be made to the algorithms presented in this article.

The signs of some of the arcs in the SDG might require plant-specific information that might not be available at the design stage. Further, the basic algorithms would not change if the SDG is used in deciding the sensor locations. Use of signs on the arcs for further differentiation of faults is discussed at the end of the article. Since the notion of signs is not used, the concepts of consistent and inconsistent cycles are also not considered. An important point to note is that the consistent and inconsistent cycles do not really change the CE behavior analysis of the process. A methodology for the optimization of different possible sensor locations in the DG representing the process is then the focus of this article.

Problem definition

The foundation of any fault-diagnosis technique lies in generating *a priori* knowledge about process faults. All process faults have to be defined clearly along with their tolerance limits. Also, the level of fault abstraction must be specified for better resolution of fault origins. For example, a tank leakage can either be defined as a separate fault or else can be further resolved into its causes, such as corrosion or chemical effects. Once all the process faults are defined, the next problem is observing all these faults. Here, two ideas—observability and fault resolution—can be introduced. Observability refers to the condition that every fault defined for the process has to be observed by at least one sensor. This would ensure that no fault goes unobserved when a given set of sensors is located on the DG. This is referred to as the ''observability condition.'' Resolution refers to the ability to identify the exact fault that has occurred. The maximum resolution that can be attained depends on the topology of the DG.

Hence, given the constraints on measurement points, the problem of resolution is generating sensor locations so that every fault is resolved to the maximum extent possible. This condition is referred to as the "highest fault resolution."

As we have seen, for a fault in the system, there exists a rooted digraph and a corresponding root node. This rooted digraph involves all the consistent branches through which the fault propagates before terminating at some node or nodes. Hence for a particular number of process faults, there are an equal number of possible rooted digraphs in the DG. Since a rooted digraph passes through various valid nodes, there is the possibility that different rooted digraphs have some common valid nodes. It is obvious that in a connected graph, some, if not all, of the rooted digraphs have common valid nodes through which they pass independently. This means that two or more faults can affect the same variable in the process. Hence, by keeping a sensor on such a node the fault symptoms of all connected root nodes can be observed.

Definition of Key Component. The problem of sensor location in order to observe all the faults now reduces to that of finding a set of nodes that would have a directed path from all the root nodes. For convenience, these nodes are referred to as the "key components." For example, in Figure 1a nodes $C7$ and $C8$ would form a set of key components that would have a directed path from every root node. Again, the set $C6$, $C7$, $C8$ would also form a set of key components. Hence, of all the sets of key components, there could be a minimum set with a minimum number of key components. This set would contain the minimum number of sensors that would satisfy the observability condition. In the next section, we solve the problem of finding a set of sensors that would observe all the faults in any given process DG, and then the problem of finding a minimum set of sensors will be addressed.

Sensor Location and Fault Observability

Problem 1: Sensor Location for Fault Observability. Given a process DG, the sensor location problem for observability is one of finding a set of nodes that is connected to all the nodes with only output arcs (root nodes).

The basic idea that is used in solving the problem is based on the following claim.

Claim 1. In a DG that is weakly connected (i.e., the corresponding undirected graph is connected) with no cycles, there is at least one directed path from a root node (node with only output arcs) to one of the nodes with only input arcs.

Proof of Claim 1. Consider a node with only output arcs. Consider the longest directed path from the node to some other node in the DG. Now, the last node in the directed path should be a node with only input arcs, otherwise it would not be the longest path. Hence every root node is connected through a directed path to one of the nodes with only input arcs. It is clear that Claim 1 is valid only for DG with no cycles.

Given a process DG, the observability problem is solved through the following sequence of steps.

1. As a first step in the algorithm, a cycle is located. All the nodes in the cycle are collapsed into a supernode.

2. This procedure is repeated until there are no more cycles in the DG. This would make the DG acyclic. For the digraph of Figure 1, the reduced digraph is shown in Figure 3.

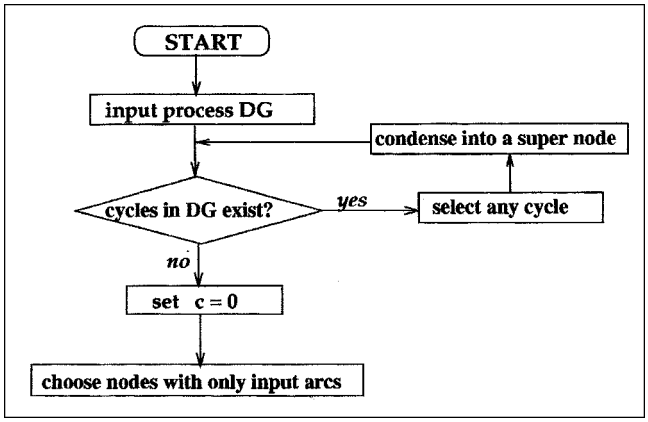

Figure 2. Method to obtain DG with only input arcs (key nodes).

3. Choose all the nodes with only input arcs. This results in a set of sensors that can observe all the faults (Claim 1). If one of the chosen nodes happens to be a supernode, then any sensor that is a part of the supernode can be used.

This set of sensors would ensure the observability condition, but the set might not be minimal. For convenience let us refer to this set as the "observability set." A flow chart of this approach is given in Figure 2, and its application to an example is given in Figure 3. Figure 3a shows the original directed graph. There are two cycles consisting of nodes $N9$, $N10$ and nodes $N11$, $N12$. Figure 3b shows the DG after the two cycles have been collapsed to supernodes $SN2$ and $SN1$, respectively. Now the observability set is given by the nodes with only input arcs. In this case, the observability set is [$C6$, $C7$, $C8$]. An important point is that the DG has to be acyclic for Claim 1 to be valid. If the DG is not acyclic, then the nodes with only input arcs need not give the observability set. This is illustrated through Figure 4. In Figure 4 the nodes with only input arcs are [$C6$, $C7$]. Clearly, these nodes would not give the observability set, as root node $R1$ cannot be observed with these two sensors. Now, if we collapsed nodes $N13$ and $N12$ into a single node, then this supernode would also become a node with only input arcs. Now, the observability set could be either [$C6$, $C7$, $N12$] or [$C6$, $C7$, $N13$].

Problem 2: Minimum Set of Sensors Ensuring Observability. The problem of generating a minimal set of sensors is the problem of picking a minimal subset of sensors from the observability set that would have at least one directed path from every root node.

This is a hard problem to solve exactly. Clearly, enumeration is a way of exactly solving the problem, but with an increasing number of key components and root nodes, the problem will become combinatorially complex. For small problems, one could generate an exact solution by checking for all possible combinations. When one tries to solve the sensor-location problem at the flow-sheet level, then enumeration would become infeasible. In this article, we first propose a greedy search (Algorithm 1) for solving the problem, and later modify the algorithm to remove redundant key

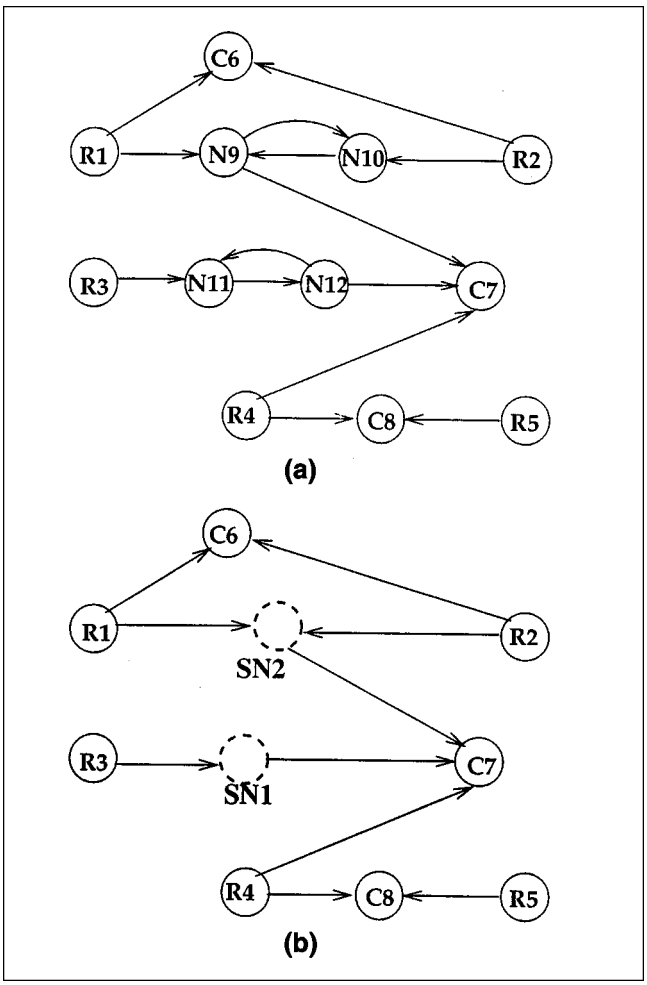

Figure 3. Reduction of digraph: (a) the original diagraph; (b) digraph with only supernodes.

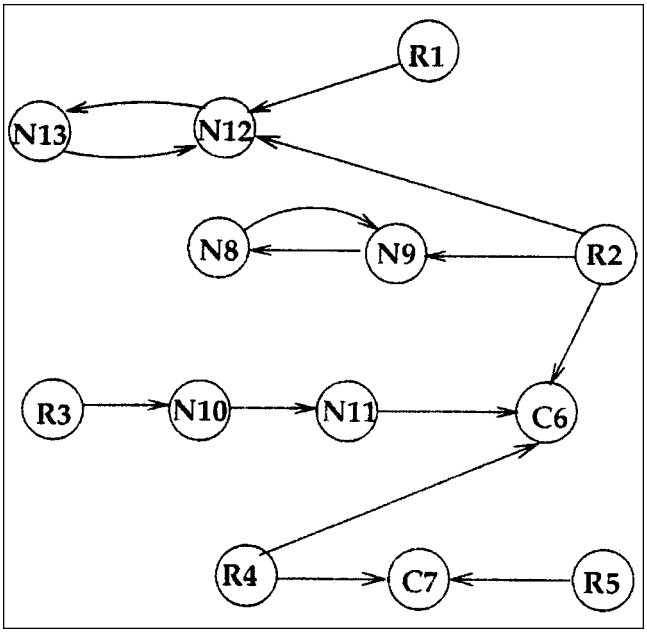

Figure 4. Explanatory figure for Claim 1.

components generated from the greedy search (Algorithm 2). We have given the solution to the problem as two algorithms to make it more understandable to the reader. In most cases, algorithm 2 will provide the actual minimum number of sensors, but is not guaranteed.

For the problem of determining the minimum number of sensors, first draw a bipartite graph between the key components and the root nodes. A bipartite graph is one whose vertex set can be partitioned into two sets in such a way that each edge joins a vertex of the first set to a vertex of the second set (Deo, 1997). There are a few methods of fault diagnosis based on bipartite graphs themselves. Since the focus of this work is on sensor location, we do not attempt to provide a detailed review of these methods. The interested reader is referred to Peng and Reggia (1990).

Whenever there is a directed path from a root node to a key component, an arc from that root node to the key component is drawn in the bipartite graph. Now the problem is one of choosing the minimum number of sensors (key components) that would cover all the root nodes. This is the well-known ''minimum set covering problem'' (Parker and Rardin, 1988). All the root nodes are said to be ''covered'' if a directed path exists from every root node to at least one of the key components. In the greedy search (Algorithm 1), the key component in the bipartite graph that has the maximum number of arcs incident on it is chosen. A check is made to determine if all the root nodes are covered by the chosen component. If they are covered, then this gives the minimum number of sensors for observability. If some of the root nodes are not covered with this key component, then all the arcs from already covered root nodes to other key components are deleted. After deletion of the arcs, the key component with the maximum number of arcs incident on it is chosen again. At this point, a check is made to see if all the root nodes are covered. This procedure is continued until all the root nodes are covered. The preceding algorithm is given in Figure 5 in terms of a flow chart. In the flow chart c is the total number of nodes chosen and C is the actual set of nodes chosen. An example illustrating the algorithm is given in Figure 6. Figure 6a shows the bipartite graph for the original digraph given in Figure 3a.

Figure 3b clearly shows that nodes [$C6$, $C7$, $C8$] form the ''observability set,'' as these are the nodes in the DG with only input arcs. It is also clear that by placing sensors on these nodes, one could detect all the faults. Now the set [$C6$, $C7$, $C8$] is not the minimal set. The minimal set is, in fact, [$C7$, $C8$]. Figures 6a and 6b show how Algorithm 1 generates the minimum number of sensors from the observability set. As a first step in the algorithm, node $C7$ is chosen as the key component, as it has the maximum number of arcs incident on it. In the next step, all the arcs other than the arcs from $R5$ to $C8$ are cut (Figure 6b) because all the other root nodes are already covered by $C7$. These arcs are shown as dotted lines in the figure. Node $C7$ is marked, as this node is already chosen. Now $C8$ is chosen as the next key component, and with this it is clearly seen that all the root nodes are now covered. This, then, is the minimal observability set, as given by Algorithm 1.

It can also be seen that this algorithm need not give the actual minimum number of sensors. To explain this, let us consider Figure 7. Using Algorithm 1, $C1$ would be chosen as

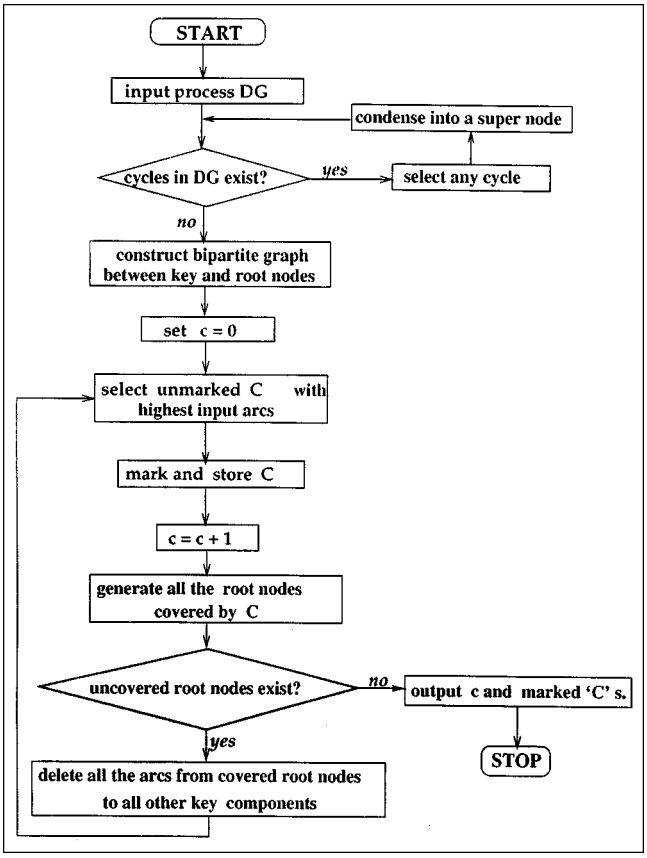

Figure 5. Flow chart for Algorithm 1.

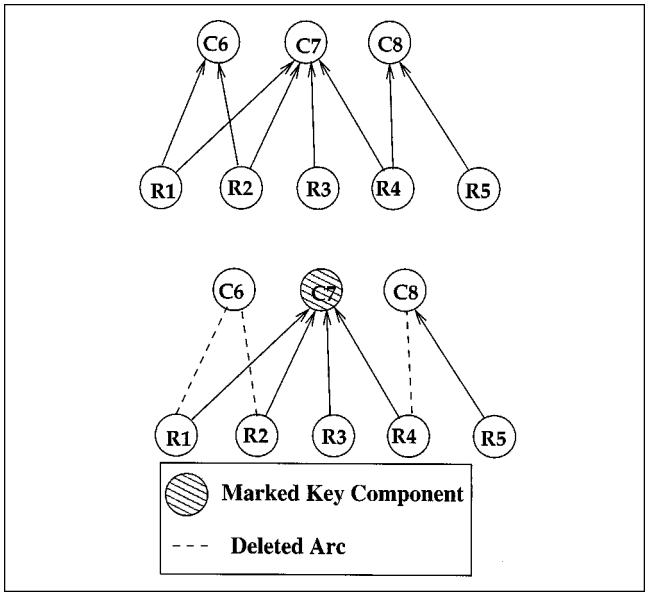

Figure 6. Conversion of Figure 3b into a bipartite graph: (a) bipartite graph; (b) bipartite graph during Algorithm 1.

the first key component, as it has the maximum number of arcs incident on it. $C1$ covers the root nodes [$f1$, $f2$, $f3$, $f4$]. As the second step, the arcs from [$f1$, $f2$, $f3$, $f4$] to [$C2$, $C3$, $C4$, $C5$] are removed. One still has to choose [$C2$, $C3$, $C4$, $C5$] to cover root nodes [$f5$, $f6$, $f7$, $f8$]. Using Algorithm 1, the minimal set identified is therefore [$C1$, $C2$, $C3$, $C4$, $C5$]. But clearly, the minimal set is actually [$C2$, $C3$, $C4$, $C5$]. The presence of a redundant component, $C1$, makes the observability set nonminimum. Though the node $C1$ was chosen first, the subsequently chosen sensor nodes among themselves cover all the root nodes covered by $C1$. Hence the key component becomes redundant in the observability set. To solve this problem, a backtracking procedure is used in the algorithm to facilitate identification and removal of the key component.

We introduce backtracking in Algorithm 2 to remove redundant key components. In this algorithm, the key component with the maximum number of arcs incident on it is chosen first and marked. All the arcs from the root nodes covered by the selected key component to all the previously marked key components are deleted. All the other arcs from the root nodes covered by the chosen key component to unmarked key components are stored in a buffer. Now, the key component is selected again based on the maximum number of incident arcs. The number of arcs that are incident on a key component is taken to be the difference between the actual number of arcs incident on the key component and the arcs incident on the key component that are stored in the buffer. This procedure is continued until all the root nodes are covered. At the end of the algorithm, all the marked key components that have no arcs incident on them are removed from the minimal set. This ensures the removal of the redundant key components. The algorithm is given in Figure 8 in terms of a flow chart.

The algorithm is explained using the example given in Figure 7. For the sake of clarity, let us denote an arc from f_i to c_j as a_{ij}. As a first step in the algorithm component $C1$ is chosen and marked. Arcs [a_{12}, a_{23}, a_{34}, a_{45}] are kept in the buffer. Now all the other key components have one (two actual arcs minus one arc in the buffer) arc incident on them. Let us choose $C2$. Root nodes f_1 and f_5 are covered by this sensor. At this stage arc a_{11} is deleted, as it is connected to an already marked key component from f_1, which is a root node covered by the currently chosen key component ($C2$). Similarly arcs a_{21}, a_{31}, a_{41} are deleted when components $C3$, $C4$, and $C5$, respectively, are chosen. At the end of the algorithm, nodes $C1$, $C2$, $C3$, $C4$, $C5$ would be chosen as the

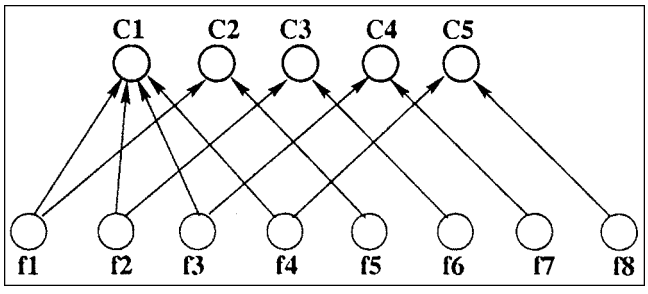

Figure 7. Example figure for Algorithm 1.

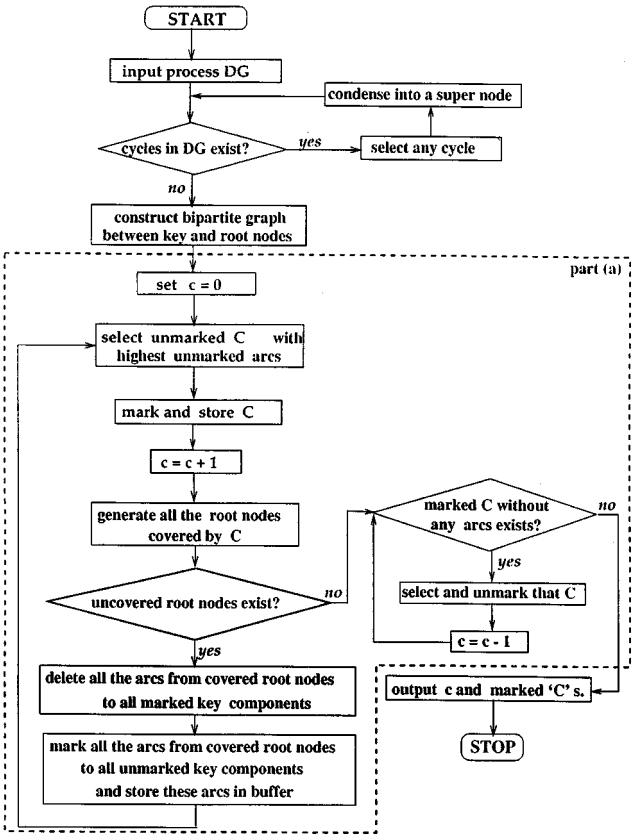

Figure 8. Flow chart for Algorithm 2.

minimal set. Now, since node $C1$ has no arcs incident on it, this node would be deleted. Hence set $[C2, C3, C4, C5]$ would be identified as the minimal set. It is obvious that though we remove key components at the end of the algorithm, observability is always ensured.

Claim 2. Observability of the faults is ensured by the set of key components generated by Algorithm 2.

Proof of Claim 2. The algorithm terminates only after all the root nodes are covered. The only way in which observability could be lost is in the removal of a key component. Clearly, observability could be lost only for the root nodes covered by the deleted key components. For a key component to be deleted, all the arcs incident on it have to be deleted. An arc from a root node to a component is deleted only if the root node is covered by some other component. Similarly, for all the arcs deleted there would be some other key component covering the corresponding root node. Hence observability is ensured.

Sensor Location and Fault Resolution. The minimum requirement for a fault-monitoring system is that it ensure observability. In the previous section, this problem was formulated and solved. For a fault-monitoring system to be useful practically, it should not only be able to observe all the faults but also resolve them to the maximum extent possible. The resolution is of course restricted by the topology of the digraph and the position of the fault or root nodes in the digraph. Also, the assumption of single-fault or multiple-fault would lead to different resolutions. In this section, the problem of sensor locations for maximum resolution under single-fault and multiple-fault assumptions is formulated and solved.

The problem of sensor location for maximum resolution under single-fault assumption could be formally stated as follows:

Problem 3: Sensor Location for Maximum Fault Resolution under Single-Fault Assumption. Let A_i be all the nodes connected to root node i, $O = \bigcup_i A_i$. Choose a set of nodes from O, denoted C. Let the nodes in C connected to root node i be denoted by C_i. The problem now is to choose a minimal set C such that $C_i \neq C_j$ whenever $A_i \neq A_j$ and at least one element from every A_i is a member of C.

Solution to Problem 3. Clearly, in the solution to the single-fault problem, two faults become indistinguishable if and only if $A_i = A_j$. Whenever they are not equal, there would be at least one sensor that could be used to distinguish the faults. A minimum number of such sensors is chosen to perform fault detection. The solution to Problem 3 would give a set of sensors that would be adequate to perform fault diagnosis, assuming there was only a single fault. To understand this, consider the following example.

Example 1. Let there be three fault classes f_1, f_2, and f_3. Now let $A_1 = [S_1, S_5]$, $A_2 = [S_2, S_3]$, and $A_3 = [S_1, S_3, S_4]$. Clearly, if only one fault is expected to occur at a time, then the set $[S_1, S_3]$ would be adequate to distinguish between the three faults. Here $C_1 = [S_1]$, $C_2 = [S_3]$, and $C_3 = [S_1, S_3]$. In the solution for a single-fault assumption, C_1, C_2 are therefore subsets of C_3, but not equal to C_3. A solution to Problem 3 would provide the optimum number of sensors that need to be used to get the maximum resolution possible among the different faults. Observability of all the faults is ensured by the condition that at least one element from every A_i is a member of C.

Problem 3 without the observability condition (the condition that at least one element from every A_i is a member of C) can be transformed into an observability problem and solved using Algorithm 2. The procedure for this transformation is as follows:

1. Define $B_{ij} = B_{ji} = A_i \cup A_j - A_i \cap A_j$. There are $n \times (n-1)/2$ such sets generated.
2. Denote each $B_{ij} = B_{ji}$ by a node. Draw a bipartite graph between this node and the nodes in the set $B_{ij} = B_{ji}$. Repeat this procedure for all the new nodes generated.
3. Without the observability condition, the minimum set of nodes that would cover all the root nodes in the bipartite graph is the solution to Problem 3.

To understand why this procedure solves the single-fault assumption problem, consider Figure 9. The figure shows the set of sensors that would be present in the set $B_{ij} = B_{ji}$ as defined before. Now if the root node $B_{ij} = B_{ji}$ is covered, then faults A_i and A_j can be differentiated. This is because every sensor in the set is either in A_i or in A_j and not in both. If this procedure is repeated for all the faults pairwise, then all the faults could be distinguished except for the ones that affect the identical set of sensors. Our aim here is to choose a minimal set of sensors that would perform single-fault identification. If we consider Example 1, three new sets will be generated. They are: $B_{12} = B_{21} = [S_1, S_2, S_3, S_5]$, B_{23}

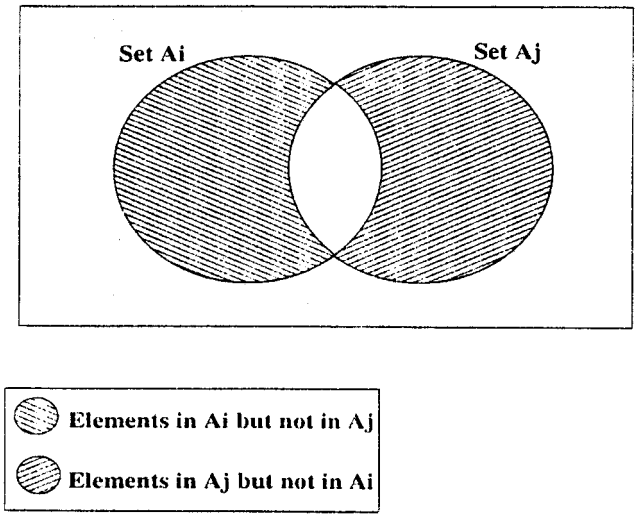

Figure 9. Explanatory Venn diagram.

$= B_{32} = [S_1, S_2, S_4]$, $B_{13} = B_{31} = [S_3, S_4, S_5]$. It is clear that sensors $[S1, S3]$ would be adequate to observe these nodes and hence to perform single-fault identification.

Algorithm 2 can be applied to this bipartite graph to determine the minimum number of nodes that would observe all B_{ij}. This set of sensors would be the set that would give maximum resolution, and in most cases would be a minimum set as well. A flow chart of the preceding algorithm (Algorithm 3) is given in Figure 10.

The only aspect of the formulation that has not been considered in this solution is the requirement of the presence of at least one element from every A_i in C. This can easily be handled by also adding A_i as root nodes in the bipartite graph and determining the minimum number of sensors required. Adding A_i to the root nodes is crucial since the algorithm would not guarantee observability otherwise, as can be seen from Claim 3 in a specific case.

Claim 3. The observability of fault i is not ensured by Algorithm 3, with B_{ij} only as the root nodes, if $A_i \subseteq A_j \forall j \neq i$.

Proof of Claim 3. It is clear that no element from A_i would be present in the sets B_{ij}. Now the observability of fault i could still be ensured only if some element of A_i is present in some other set $B_{k,l}$, where $k, l \neq i$. This is not possible because $A_i \subseteq A_k$, A_l, and hence $B_{kl} \cap A_i = \phi \forall k,l \neq i$. Hence the observability of fault i cannot be ensured.

Even if $A_i \not\subset A_j \forall j \neq i$, it can still be easily shown that the observability of some faults might not be ensured. Hence, A_i should also be added as root nodes in the algorithm. If we also add A_i as root nodes, then the observability of fault i is always ensured even if $A_i \subseteq A_j \forall j$.

Assuming multiple faults, the sensor-location problem can be solved as an extension of the single-fault assumption problem. Let us consider a specific case where single faults and two simultaneous faults are important. This problem can be solved through the following series of steps:

1. Define $A_{ij} = A_{ji} = A_i \cup A_j$. There are ${}^nC_2$ new nodes generated. Mark all these as root nodes along the n nodes A_i. There are a total of ${}^nC_2 + n$ root nodes in the problem, with the corresponding sets of sensors that they affect. Let us call this System 1.

2. Solve the single-fault-assumption problem for System 1. This will give the solution to the problem just posed.

In the solution to the double-fault case, some redundant sets clearly might be generated, and one could considerably reduce the number of root nodes for which the observability problem has to be solved. We do not attempt this here. Instead, we simply point out how the single-fault-assumption solution can be used to handle the double-fault case also. In fact, this approach provides us with a framework for posing and solving various kinds of sensor-location problems. This is a more important aspect of the proposed solution strategy. To illustrate this point, consider the following. In a typical plant scenario, one might not be concerned with all multiple- or double-fault possibilities. There might be some faults that might have a high probability of occurring together. One could simply add such sets of faults to the original single-fault sets and get a solution to the sensor-location problem. This is an important aspect of the algorithm that gives the designer the ability to pose and solve relevant sensor-location problems.

Illustrative Example. To illustrate the concepts developed, fault diagnosis of the digraph of Figure 3b is performed. The

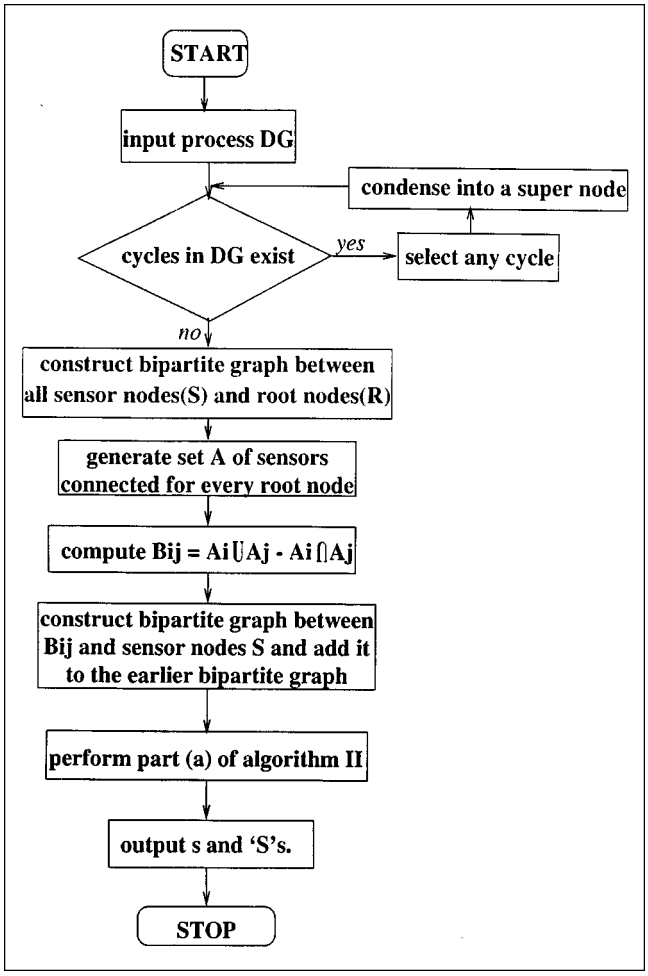

Figure 10. Flow chart for Algorithm 3.

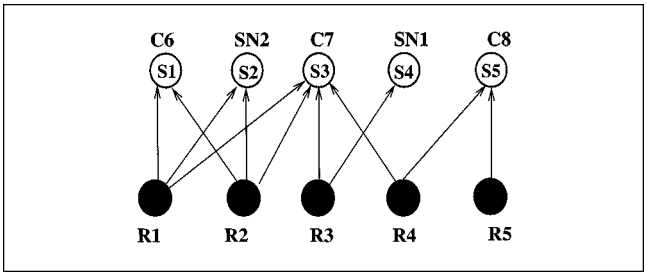

Figure 11. Root and sensor node bipartite graph for Figure 3b.

root and sensor node bipartite graph for this digraph is shown in Figure 11. Algorithm 2 is applied to get the set of sensors just for observability. This set turns out to be [*S*3, *S*5] (Figure 11).

Algorithm 3 is used for fault diagnosis under the single-fault assumption. As a first step, the sets $B_{ij} = A_i \cup A_j - A_i \cap A_j$ are constructed for all i and j. Set A of the sensors for different root nodes is given in Table 1. The root nodes that affect the same sensor set have been combined as a single root and they cannot be distinguished from one another. For example, nodes $R1$ and $R2$ have the same sensor set, and have been combined and denoted by the same root node $A1$. The set B_{ij} for the faults are shown in Table 2. The faults consist of the original faults plus the set B_{ij}. Algorithm 2 is applied on this extended fault system and the sensor set is found to be [*S*1, *S*3, *S*5]. Faults 1 and 2 are indistinguishable, since the A's for both faults are [*S*1, *S*2, *S*3]. All other faults can be distinguished from each other by the chosen sensor set, and this can be easily verified.

Under the double-fault assumption, first the sets $A_{ij} = A_i \cup A_j$ are constructed. These sets are shown in Table 3. Each A_{ij} is also considered to be a fault. The faults are thus the original faults (A_i) and the new A_{ij}. Call this System 1. Algorithm 3 (single-fault algorithm) is now applied to this system and the set of sensors, [*S*2, *S*3, *S*4, *S*5], is obtained.

Table 1. Set A of Sensors for Different Root Nodes in Figure 11

R Nodes	S Nodes	Set A
$R1$, $R2$	[*S*1,*S*2,*S*3]	$A1$
$R3$	[*S*3,*S*4]	$A2$
$R4$	[*S*3,*S*5]	$A3$
$R5$	[*S*5]	$A4$

Table 2. Set B_{ij} for Single-Fault Case for A's of Table 1

Set B	Sensor Nodes
$B12$	[*S*1,*S*2,*S*4]
$B13$	[*S*1,*S*2,*S*5]
$B14$	[*S*1,*S*2,*S*3,*S*5]
$B23$	[*S*4,*S*5]
$B24$	[*S*3,*S*4,*S*5]
$B34$	[*S*3]

Table 3. Set A_{ij} for Multiple-Fault Case for A's of Table 1

Set A_{ij}	Sensor Nodes
$A12$	[*S*1,*S*2,*S*3,*S*4]
$A13$	[*S*1,*S*2,*S*3,*S*5]
$A14$	[*S*1,*S*2,*S*3,*S*5]
$A23$	[*S*3,*S*4,*S*5]
$A24$	[*S*3,*S*4,*S*5]
$A34$	[*S*3,*S*5]

Clearly, the number of sensors required for the double-fault diagnosis is more than for the single-fault case. For System 1, indistinguishable faults also can be identified, just as in the single-fault assumption case. For example, from Table 3, $A_{2,3} = A_{2,4}$. This means that the occurrence of $A2$ and $A3$ together cannot be distinguished from the occurrence of $A2$ and $A4$ together. From Table 1, we see that $A2$ corresponds to fault $R3$, $A3$ to $R4$, and $A4$ to $R5$. Hence, if they occur together, faults $R3$ and $R4$ cannot be distinguished from the simultaneous occurrence of faults $R3$ and $R5$.

The application of the algorithms developed on CSTR and FCCU case studies are presented next.

CSTR Case Study (*Mohindra and Clark, 1993*). The CSTR system shown in Figure 12 is considered. Its corresponding digraph is shown in Figure 13. Mohindra and Clark (1993) considered this system for analyzing their improved fault-diagnosis algorithm. As a first step, the cycle in DG given in Figure 13 (nodes 21, 22, 24) is identified and replaced by a supernode (node 21). In this DG, nodes 1–13 are the root nodes (corresponding to faults), as they have no input arcs. Nodes 14 and 16 are the key components, as they have no output arcs. Nodes 14–23 are nodes on which sensors can be placed. Algorithm 2 is applied to find the sensors required to observe all the faults. This gives node 14 as the sensor node. Hence, just one sensor is required to meet only the observability condition and this should be placed on node 14. It is to be noted that in this case, node 16 can also be a sensor node, because it is connected to all the process faults. Another important observation is that by placing the sensor on

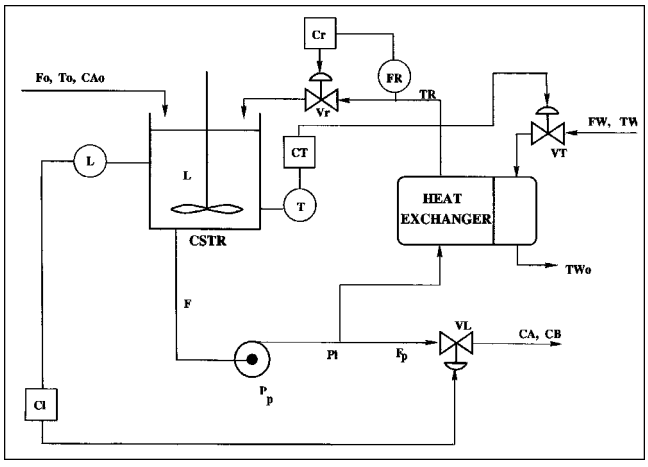

Figure 12. Process digram of a CSTR system.

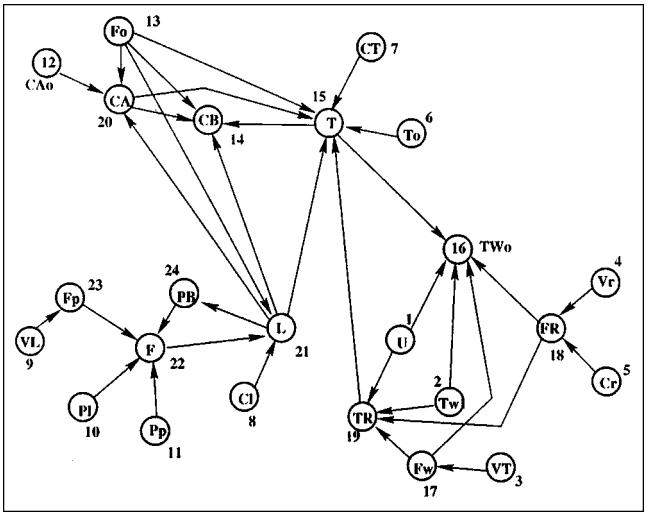

Figure 13. Process digraph of a CSTR system.

Table 4. Set A of Sensors for Root Nodes in Figure 13

Root Nodes (R)	Sensor Nodes (S)	Set A
$R1, R2$	[$S14, S15, S16, S19$]	$A1$
$R3$	[$S14, S15, S16, S17, S19$]	$A2$
$R4, R5$	[$S14, S15, S16, S18, S19$]	$A3$
$R6, R7$	[$S14, S15, S16$]	$A4$
$R8, R10, R11, R13$	[$S14, S15, S16, S20, S21$]	$A5$
$R9$	[$S14, S15, S16, S20, S21, S23$]	$A6$
$R12$	[$S14, S15, S16, S20$]	$A7$

either node 14 or 16, all the faults can be detected, but none of them can be distinguished from any other.

Algorithm 3 is applied to obtain the set of sensors that will give maximum resolution under the single-fault assumption case. Sets A associated with the faults are given in Table 4. Corresponding to these A_i, the sets $B_{ij} = A_i \cup A_j - A_i \cap A_j$ are constructed and are shown in Table 5. Now each B_{ij} is represented as a root node, and a bipartite graph is constructed between these nodes and the sensor nodes. This bipartite graph is added to the earlier bipartite graph, and part (a) of Algorithm 2 is applied to this new system. This gives nodes [$S15$, $S17$, $S18$, $S19$, $S20$, $S21$, $S23$] as the sensor set. The detailed results are tabulated in Table 6. We can again verify that the given sensors would be able to perform single-fault identification of maximum resolution.

FCCU Case Study (Mylaraswamy et al., 1994). As a case study, sensor location on a complex FCCU unit shown in Figure 14 is performed. There are a total of 43 nodes in this

Table 5. Set B_{ij} for Sensor Sets in Table 4

Set B	Sensor Nodes	Set B	Sensor Nodes
$B12$	[$S17$]	$B34$	[$S18, S19$]
$B13$	[$S18$]	$B35$	[$S18, S19, S20, S21$]
$B14$	[$S19$]	$B36$	[$S18, S19, S20, S21, S23$]
$B15$	[$S19, S20, S21$]	$B37$	[$S18, S19, S20$]
$B16$	[$S19, S20, S21, S23$]	$B45$	[$S20, S21$]
$B17$	[$S19, S20$]	$B46$	[$S20, S21, S23$]
$B23$	[$S17, S18$]	$B47$	[$S20$]
$B24$	[$S17, S19$]	$B56$	[$S23$]
$B25$	[$S17, S19, S20, S21$]	$B57$	[$S21$]
$B26$	[$S17, S19, S20, S21, S23$]	$B67$	[$S21, S23$]
$B27$	[$S17, S19, S20$]		

flow sheet, of which 16 are the fault nodes. In general, all the variables involved in the process cannot be measured. So, of the remaining 27 nodes, only some are possible sensor nodes. Hence, a reduced digraph consisting only of root nodes and measurable nodes is generated, as shown in Figure 15. Now, fault diagnosis is performed on this digraph. For observability, Algorithm 2 is applied to this flow sheet. This gives nodes $S3$ and $S4$ as the sensor nodes. Hence, the set of sensors just for observability is [$S3$, $S4$]. It is worth noting that this set is the minimal set of sensors, which ensures observability of all faults.

For fault diagnosis under the single-fault assumption, Algorithm 3 is applied to Figure 15. Table 7 lists the A set associated with the root nodes. As illustrated in Figure 10, in this algorithm sets $B_{ij} = A_i \cup A_j - A_i \cap A_j$, which are associated with all faults, are constructed first. Table 7 lists the A_i sets associated with the root nodes, and Table 8 gives the B_{ij} sets. For the sake of simplicity, only the distinct As and B_{ij} are shown. So, now the root nodes are the original 16 root nodes plus the $16 \times 15/2$ new nodes. Now, Algorithm 2 is applied to this system. This gives the set [$S1$, $S4$, $S5$, $S6$, $S7$] as the set of sensor nodes. These sensor positions are marked in Figure 15. Also, for the two faults i and j, if $B_{ij} = \phi$, then faults i and j cannot be distinguished from each other. For example, for faults $R11$ and $R15$, $A_{11} = A_{15} =$ [$S1$, $S2$, $S3$, $S4$, $S5$, $S6$, $S7$, $S8$, $S9$]. Hence, $B_{11,15} = \phi$ and faults $R11$ and $R15$ cannot be distinguished from each other. A complete list of indistinguishable sets of faults for the present case is given in Table 9.

For the double-fault case, solving the sensor-location problem is done by assuming that two simultaneous faults can occur along with the possibility of the occurrence of a single fault. Under this assumption, the sets $A_{ij} = A_i \cup A_j$ are formed for all faults i and j. Here, since we have 16 faults, we get ${}^{16}C_2$ sets. Each A_{ij} is treated as a root node along with the original root nodes. Now, Algorithm 3, which was

Table 6. Results of Sensor Location Algorithm for CSTR Using Tables 4 and 5

Details	Observability of All Faults	Single-Fault Assumption
Sensor positions	$S14$	$S15, S17, S18, S19, S20, S21, S23$
Number of sensors	$c = 1$	$s = 7$
Indistinguishable set I	All faults are indistinguishable	[$R1, R2$],[$R3$],[$R4, R5$] [$R6, R7$],[$R8, R10, R11, R13$] [$R9$],[$R12$]

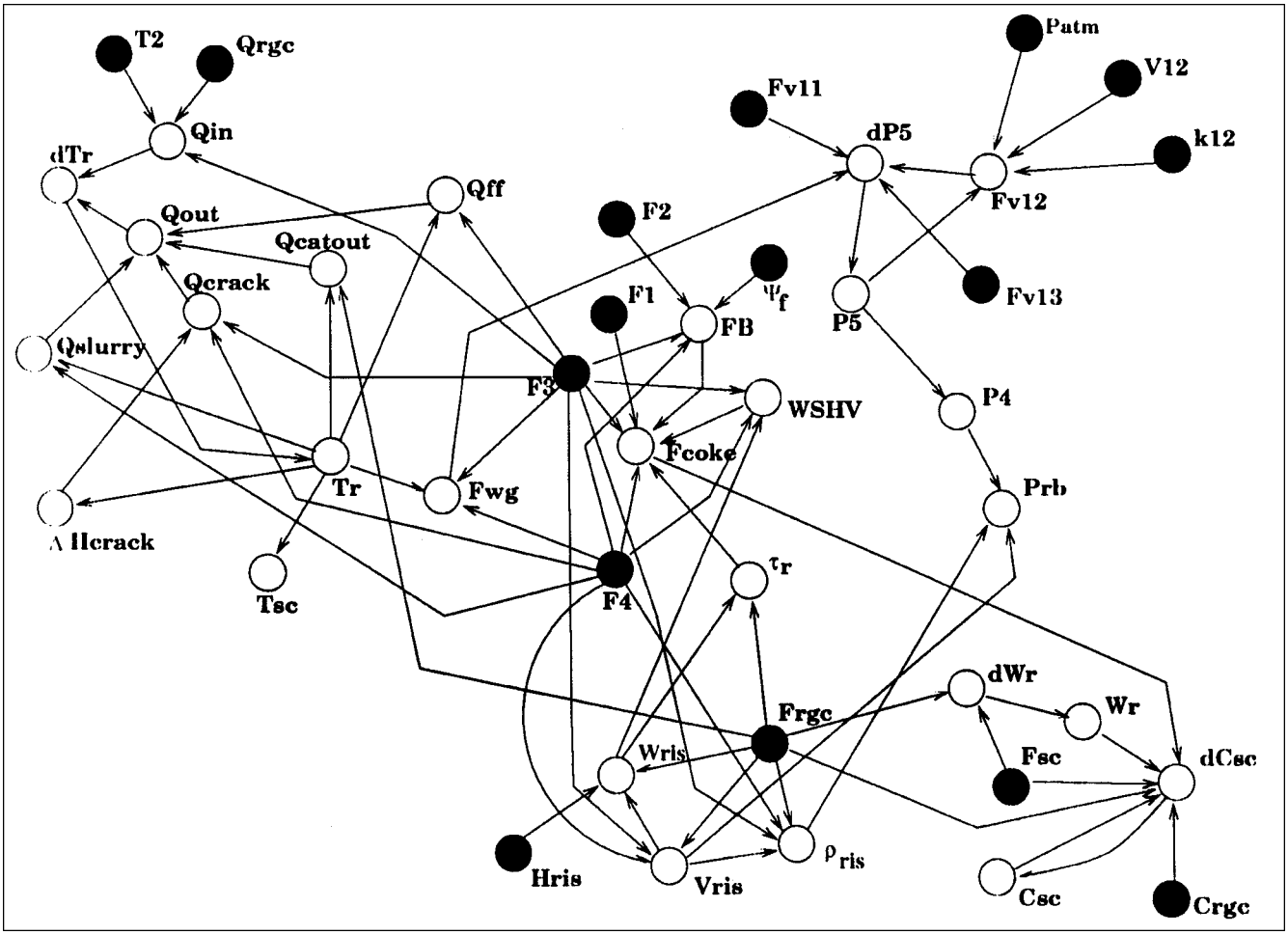

Figure 14. FCCU reactor digraph.

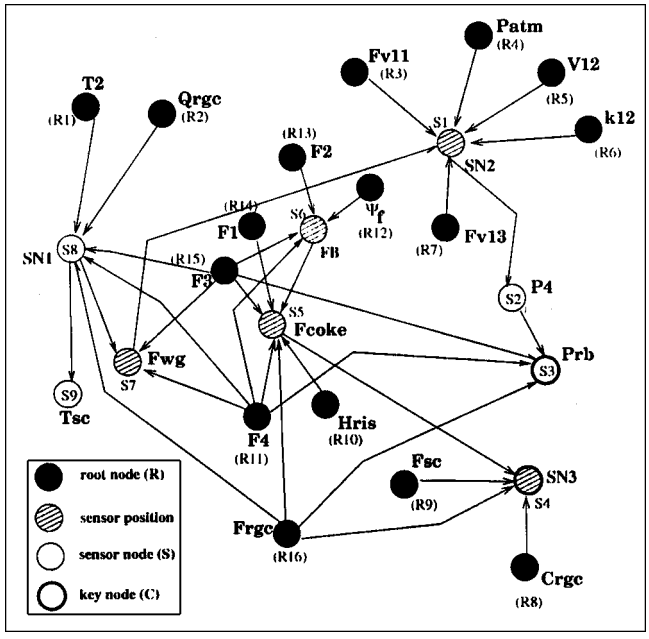

Figure 15. Reduced FCCU digraph with sensor nodes and root nodes.

Table 7. Set A of Sensors for Root Nodes in Figure 15

Root Nodes (R)	Sensor Nodes (S)	Set A
$R8, R9$	$[S4]$	$A1$
$R10, R14$	$[S4, S5]$	$A2$
$R12, R13$	$[S4, S5, S6]$	$A3$
$R1, R2$	$[S1, S2, S3, S7, S8, S9]$	$A4$
$R16$	$[S1, S2, S3, S4, S5, S7, S8, S9]$	$A5$
$R11, R15$	$[S1, S2, S3, S4, S5, S6, S7, S8, S9]$	$A6$
$R3, R4, R5, R6, R7$	$[S1, S2, S3]$	$A7$

Table 8. Set B_{ij} for Sensor Sets in Table 7

Set B	Sensor Nodes	Set B	Sensor Nodes
$B12$	$[S5]$	$B34$	$[S1, S2, S3, S4, S5, S6, S7, S8, S9]$
$B13$	$[S5, S6]$	$B35$	$[S1, S2, S3, S6, S7, S8, S9]$
$B14$	$[S1, S2, S3, S4, S7, S8, S9]$	$B36$	$[S1, S2, S3, S7, S8, S9]$
$B15$	$[S1, S2, S3, S5, S7, S8, S9]$	$B37$	$[S1, S2, S3, S4, S5, S6]$
$B16$	$[S1, S2, S3, S5, S6, S7, S8, S9]$	$B45$	$[S4, S5]$
$B17$	$[S1, S2, S3, S4]$	$B46$	$[S4, S5, S6]$
$B23$	$[S6]$	$B47$	$[S7, S8, S9]$
$B24$	$[S1, S2, S3, S4, S5, S7, S8, S9]$	$B56$	$[S6]$
$B25$	$[S1, S2, S3, S7, S8, S9]$	$B57$	$[S4, S5, S7, S8, S9]$
$B26$	$[S1, S2, S3, S6, S7, S8, S9]$	$B67$	$[S4, S5, S6, S7, S8, S9]$
$B27$	$[S1, S2, S3, S4, S5]$		

Table 9. Results of Sensor Location Algorithm for FCCU Using Table 8

Details	Observability of All Faults	Single-Fault Assumption
Sensor positions	$S3, S4$	$S1, S4, S5, S6, S7$
Number of sensors	$c = 2$	$s = 5$
Indistinguishable set I	$[R1, R2, \ldots, R7]$	$[R1, R2], [R8, R9]$
	$[R11, R15, R16]$	$[R3, R4, \ldots, R7], [R10, R14]$
	$[R8, R9, R10, R12, R13, R14]$	$[R11, R15], [R12, R13], [R16]$

developed for the single-fault assumption, is applied to this system. This gives the set [*S*1, *S*4, *S*5, *S*6, *S*7] as the set of sensor nodes. As in the single-fault case, indistinguishable faults can be identified here also. For example, A, CA_2, and therefore the occurrence of fault 10 only cannot be distinguished from the simultaneous occurrence of faults 8 and 10. An important and interesting point to note here is that the minimum number of sensors for performing single- and double-fault identification with maximum resolution turns out to be the same. This would mean that resolution for double-faults cannot be improved with more sensors. This is a result that is not obvious from the DG of the process.

Further Remarks

Significance of node and branch signs

Faults that cannot be distinguished using the proposed approach can be isolated further by exploiting the inherent nature of the faults and their effects on the sensor nodes. In practice, some of the faults (root nodes) can be assumed to exhibit only one abnormal state, for example, tank leakages, catalyst deactivation, and valve choking, to name a few. Such faults are assigned only one state. Hence, if they occur, these faults always affect the corresponding sensor in only one direction, depending on the sign of the branch between them. Further, if two such faults that are indistinguishable have mutually opposite effects on the sensed variable, then the state of the sensor node itself signifies the exact origin of the fault. This implies that these faults can be resolved. Hence, greatly improved fault diagnosis can be achieved with the same number of sensors. This procedure can be extended to a pair of similar faults with only one abnormal state and are affecting different sensor nodes. These faults can now be checked for their ''path signs.'' If the faults have opposite path signs (product of all the branch signs in the path), then these two faults can be easily resolved in a similar way. For such studies, one needs to place signs on the arcs in the DG and work with that SDG.

Importance of ''sensor location optimization''

The importance of the optimization problem of sensor locations is mainly due to its ability to handle the different constraints of fault diagnosis and variable measurements. The significance of this problem is outlined below.

1. Observing the symptoms of all the possible faults with a minimum number of sensors optimizes the cost of a fault-monitoring system.

2. Using some more sensors and the inherent behavior of fault-symptom propagation to isolate possible faults from one another reduces the fault identification time. This helps in rapid corrective measures and in reducing damage.

3. From the point of view of process analysis, sensor location is very important. The optimization problem can be formulated with constraints on critical measurements, such as controlled variables. The rest of the sensors can be optimally placed in order to observe all the faults.

4. Sensor locations are also important from the point of view of safety. Crucial and dangerous faults have to be detected quickly and diagnosed. The deviation in some variable due to the occurrence of a fault, beyond a certain limit, may be dangerous to the process and may cause severe accidents. Early detection of such symptoms depends entirely on the fault-monitoring system.

5. Other interesting results, such as the evaluation of sets of sensors for better fault diagnosis, can be obtained as shown by the FCCU case study. Further, this approach forms a framework for various kinds of sensor-location studies, as pointed out in the article.

Conclusions and Future Scope

In this article, an algorithm for designing an efficient fault-monitoring system was proposed. The algorithm uses the directed graph that models the CE propagation between various process variables. The concepts of observability and fault resolution are introduced. The problem of sensor location that would ensure observability and fault resolution is formulated. Graph algorithms have been proposed to solve these problems. Simple examples have been used to illustrate and explain the proposed algorithms. A CSTR case study and a complex FCCU case study were used to demonstrate the utility of the proposed algorithm for large flow sheets also.

There are a number of enhancements that can be incorporated into the proposed algorithm. First, signs could be placed on the DG, and sensor-location problems could be solved using the SDG. The basic algorithms would not change very much, but one could conceivably achieve better fault resolution in some cases based on the signs on the arc. Instead of a purely CE analysis, which is used in both DG and SDG, one could pose the sensor-location problem based on semiqualitative, order-of-magnitude models. In fact, one could have sensor-location analysis based on the qualitative analysis at a lower level, and further enhancements could be achieved using other models at a higher level. These aspects currently are being pursued. Other important aspects that have to be considered in a comprehensive solution to the sensor-location problem are: failure probabilities; sensor failure probability; severity of particular faults; and the cost of the sensors. We are also currently pursuing avenues in which these en-

hancements can be integrated into the solution to the sensor-location problem proposed in this article.

Notation

c = number of sensors for fault observability
I = set of indistinguishable faults
N = number of root nodes in the DG
$O = \bigcup_i A_i$
S = sensor node
s = number of sensors for highest fault resolution

Literature Cited

Ali, Y., and S. Narasimhan, ''Sensor Network Design for Maximizing Reliability of Linear Processes,'' *AIChE J.*, **39**(5), 820 (1993).

Ali, Y., and S. Narasimhan, ''Redundant Sensor Network Design for Linear Processes,'' *AIChE J.*, **41**(10), 2237 (1995).

Ali, Y., and S. Narasimhan, ''Sensor Network Design for Maximizing Reliability of Bilinear Processes,'' *AIChE J.*, **42**(9), 2563 (1996).

Chang, C. C., and C. C. Yu, ''On-Line Fault Diagnosis using the Signed Directed Graph,'' *Ind. Eng. Chem. Res.*, **29**, 1290 (1990).

Chang, C. C., K. N. Mah, and C. S. Tsai, ''A Simple Design Strategy for Fault Monitoring Systems,'' *AIChE J.*, **39**(7), 1146 (1993).

Deo, N., *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall of India, New Delhi (1997).

Iri, M., K. Aoki, E. O'Shima, and H. Matsuyama, ''An Algorithm for Diagnosis of System Failures in Chemical Processes,'' *Comput. Chem. Eng.*, **3**, 489 (1979).

Kramer, M. A., and B. L. Palowitch, Jr., ''A Rule-Based Approach to Fault Diagnosis using the Signed Directed Graph,'' *AIChE J.*, **33**(7), 1067 (1987).

Lambert, H. E., ''Fault Trees for Locating Sensors in Process Systems,'' *Chem. Eng. Prog.*, 81 (Aug. 1977).

Madron, F., and V. Veverka, ''Optimal Selection of Measuring Points in Complex Plants by Linear Models,'' *AIChE J.*, **38**(2), 227 (1992).

Mohindra, S., and P. A. Clark, ''A Distributed Fault Diagnosis Method Based on Digraph Models: Steady-State Analysis,'' *Comput. Chem. Eng.*, **17**(2), 193 (1993).

Mylaraswamy, D., S. N. Kavuri, and V. Venkatasubramanian, ''Systematic Development of Causal Digraph Models for Chemical Processes,'' AIChE Meeting, San Francisco (1994).

O'Shima, E., J. Shiozaki, H. Matsuyama, and M. Iri, ''An Improved Algorithm for Diagnosis of System Failures in the Chemical Process,'' *Comput. Chem. Eng.*, **9**, 285 (1985).

Parker, R. G., and R. L. Rardin, *Discrete Optimization*, Academic Press, San Diego (1988).

Peng, Y., and J. A. Reggia, *Abductive Inference Models for Diagnostic Problem Solving*, Springer-Verlag, New York (1990).

Manuscript received May 27, 1998, and revision received Sept. 14, 1998.