Redundant Sensor Network Design for Linear Processes

Yaqoob Ali and Shankar Narasimhan
Dept. of Chemical Engineering, Indian Institute of Technology, Kanpur-208 016, UP, India

Measurements of process variables have a considerable impact on the control, optimization, safety, and reliability of chemical processes. In a 1993 article, Ali and Narasimhan developed a systematic graph-theoretic procedure for optimally locating a minimum number of sensors in linear steady-state processes. The sensor network was designed to maximize the probability of estimating variables when sensors are likely to fail. This article extends that procedure for the optimal design of a redundant sensor network for linear processes. The algorithm proposed also accounts for specifications of measurable and important variables. The efficiency and robustness of the proposed algorithm are demonstrated on realistically large processes.

Introduction

Effective control and safe operation of chemical plants require the measurements of process variables such as flow rates, temperatures, and compositions. In large process plants, hundreds of variables exist, although only a subset of these is used in routine operation.

Generally, in a chemical process only some of the variables are measured due to reasons of cost, technical feasibility, or complexity. The unmeasured variables are usually estimated by exploiting the mass and energy balance relationships between different variables. The measured variables and all unmeasured variables that can be uniquely estimated are denoted as *observable*. The observability of variables depends on the process structure as well as the locations of sensors. Sensor network design is concerned with the optimal locations of measurements so as to ensure the observability of variables and satisfy other desired objectives.

Attempts have been made in the past to tackle the problem of sensor network design for steady-state processes using the concepts of observability and estimation accuracy of process variables. Vaclavek and Loucka (1976) were the first to develop a strategy for sensor location so as to ensure the observability of a specified set of important variables in a pure mass-flow or multicomponent process. The problem of sensor placement for a linear (mass-flow) process has also been addressed by Madron and Veverka (1992), which is essentially an extension of the work of Vaclavek and Loucka (1976). Their method makes use of Gauss-Jordan elimination to identify a minimum set of variables that need to be measured in order to observe all important variables while simultaneously minimizing the overall cost of sensors. Kretsovalis and Mah (1987a) quantified the effect of sensor placement on the accuracy of estimates for mass-flow processes, and used the results to develop a combinatorial search algorithm for sensor network design.

The methods just cited do not address the issue of sensor failures and its effect on the observability of variables, nor do they take it into account in sensor placement strategies. In our previous article (Ali and Narasimhan, 1993), we tackled the problem of sensor network design for steady-state linear processes when sensors are likely to fail. We proposed the concept of reliability of estimation of a variable, which gives the probability of estimating a variable for any given sensor network and specified sensor failure probabilities. Based on this concept, a sensor network design algorithm called SENNET was developed for maximizing the least reliability among all variables. The sensor network, however, was designed using only a minimum number of sensors required to ensure observability of all variables.

In the present article we extend this approach for the design of a *redundant* sensor network. By a redundant sensor network we imply that more sensors are used than the minimum required to ensure observability of all variables. Redundancy can be introduced by measuring a variable using more than one sensor. This type of redundancy is referred to as hardware redundancy. Redundancy can also be introduced by measuring more variables than the minimum required to ensure observability, so that there are multiple ways of estimating a variable. This type of redundancy is known as spa-

Correspondence concerning this article should be addressed to S. Narasimhan.

tial redundancy (Mah, 1990). In this work, we have developed a graph-theoretic algorithm for a redundant sensor network design that maximizes the reliability of variables in a mass-flow process. Both types of redundancy (spatial and hardware) are considered in our approach. In addition, our method can take into account specifications such as measurable or important variables.

It should be noted that in previous work (Mah, 1990), the term "redundant variable" was used only for *measured variables* that can also be indirectly estimated. Although several algorithms have been developed for classifying variables as observable or redundant for a given sensor placement (Kretsovalis and Mah, 1987b, 1988a,b; Crowe, 1989), no method is currently available that quantifies the degree of redundancy of variables or that exploits this in a measurement placement strategy. Our present work attempts to bridge this gap.

In developing our approach, several new issues were tackled. First, due to the use of redundant sensors there are multiple ways of estimating each variable. By making use of the concept of cutsets of a graph, we have explicitly identified the different ways of estimating each variable. Secondly, an efficient algorithm has been developed for computing the reliability of a variable from the different ways of estimating a variable and sensor failure probabilities. Finally, we have also quantified the minimum number of sensors required to make all *important* variables observable when specifications on variables are imposed. Our design algorithm is shown to perform efficiently and give optimal or near-optimal solutions for realistically large processes.

Ways of Estimating a Variable

In our previous article (Ali and Narasimhan, 1993) we showed that the minimum number of sensors, $n_{\min}$, required to observe all variables (assuming all variables are measurable and important) is given by

$$n_{\min} = e - n + 1 \tag{1}$$

where, e is the number of edges and n is the number of nodes (including the environmental node) in the process graph.

We also showed that in this case, unmeasured variables form a spanning tree (important graph-theoretic terms are defined in Appendix A; a more detailed description is available in Ali and Narasimhan (1993)). Furthermore, there is a unique way for estimating every variable. An unmeasured variable is estimated through its fundamental cutset with respect to that spanning tree. Since there is a unique way for estimating every variable, the reliabilities can also be computed easily. When redundant (more than $n_{\min}$) sensors are used, however, none of these properties hold. The first step, therefore, is to identify all the ways of estimating a variable for any given redundant sensor network. We prove in the following theorem that all the different ways of indirectly estimating a variable can be obtained using cutsets of a graph.

Theorem 1

Given a sensor network for a pure mass-flow process, all the different ways of *indirectly* estimating mass flow of stream i are given by the cutsets that contain stream i in which the mass flow of every other stream is measured.

Proof. Let Q_i be the set of cutsets containing stream i. Let Q_i^m be the subset of Q_i such that in every cutset belonging to Q_i^m, all edges (excluding i) are measured. Since we are interested in finding all the indirect ways of estimating mass flow of i, edge i can be assumed to be unmeasured in this proof.

Corresponding to every cutset K_i^m of Q_i^m, a mass balance equation can be written involving only the mass flows of all streams in that cutset. Since every mass flow in this equation other than that of i is measured, it gives a way of indirectly estimating mass flow of i.

The question arises whether any other cutset of Q_i not belonging to Q_i^m can be used to obtain a different indirect way of estimating mass flow of i. For example, let $K_{i,j}^u$ be a cutset of Q_i in which j is the only other unmeasured stream. Let K_j^m be another cutset through which mass flow of stream j can be indirectly estimated. The question is whether we can obtain a different indirect way of estimating mass flow of stream i by combining the equations corresponding to cutsets $K_{i,j}^u$ and K_j^m. We show in the following lemma that such combinations do not provide any new way of indirectly estimating mass flow of stream i than those already given by the equations corresponding to cutsets of Q_i^m. Without loss of generality, we consider the combination of cutset $K_{i,j}^u$ that contains only one unmeasured edge j (other than i), and cutset K_j^m in which all edges except j are measured. The proof for all other combinations of more than two cutsets (required, for example, if cutset $K_{i,j}^u$ contains more than two unmeasured edges) can be obtained by recursive application of the following lemma.

Lemma 1

There exists a cutset, member of Q_i^m, which contains only a subset of edges present in the union of cutsets $K_{i,j}^u$ and K_j^m.

Proof. Let K be the ring sum of $K_{i,j}^u$ and K_j^m. Since the ring sum of two cutsets must be a subset of the union of the same two cutsets, we get:

$$K \subseteq K_{i,j}^u \cup K_j^m.$$

Moreover, j cannot be a member of K since it is common to both $K_{i,j}^u$ and K_j^m. Therefore, it follows that

$$K \subseteq K_{i,j}^u \cup K_j^m - \{j\}.$$

From the definitions of $K_{i,j}^u$ and K_j^m, we can observe that K contains only measured edges, other than i. Moreover, K must contain i, since i is assumed to be unmeasured and cannot be a member of K_j^m.

It can be proved (Deo, 1974, p. 72) that the ring sum of two cutsets gives another cutset or an edge-disjoint union of cutsets. If K is a cutset, then it must be a member of Q_i^m, because it contains i and all other edges are measured. If K is an edge-disjoint union of cutsets, then it can be decomposed into many cutsets, only one of which contains i. Clearly, this cutset must be a member of Q_i^m, because all the other edges of this cutset are measured. These conclusions prove the lemma.

The preceding lemma implies that the equation corresponding to K does not provide a new indirect way of estimating i, since it has already been accounted for by the equation corresponding to the cutset of Q_i^m, which is a subset of K.

From the preceding theorem, we find that the cutsets in Q_i^m give *all* the indirect ways of estimating a variable i. If i is a measured variable, then its direct measurement provides one additional way of estimating it. In summary, the following steps are used to obtain all possible ways of estimating mass flow of stream i.

Step 1. Obtain all cutsets of the process graph. Many different algorithms are available in the literature to evaluate all cutsets of a graph. We have implemented the algorithm of Tsukiyama et al. (1980) for this purpose.

Step 2. Pick all the cutsets in which edge i is a member, and discard those cutsets that have unmeasured edges other than i. The remaining cutsets give the mass balance equations through which mass flow of stream i can be estimated. The number of ways is simply the number of remaining cutsets. If mass flow of stream i is also measured, then the direct measurement provides one more way of estimating it.

Example 1

Consider the ammonia network shown in Figure 1. We assume that the mass flows of streams 3, 4, 5, 6, and 7 are measured. It is desired to obtain all possible indirect ways through which mass flow of stream 8 can be estimated. We evaluate all cutsets of the graph that are listed in Table 1. From the cutsets in which edge 8 is a member, we choose those in which all other edges are measured. They are

(i) {3, 6, 8}
(ii) {3, 5, 7, 8}
(iii) {4, 5, 6, 8}
(iv) {4, 7, 8}

These cutsets can be used to indirectly estimate the mass flow of stream 8. Note that by combining cutset {1, 6, 8} in which edge 1 is unmeasured and cutset {1, 3} (see Table 1) we get an equation relating edges 3, 6, and 8, through which mass flow of stream 8 can be estimated. This is, however, the same as the equation corresponding to cutset (i) just listed.

Two important observations that highlight the deficiency of the redundancy concept can be made from this illustration.

1. We note in the preceding example that stream 8, which is an unmeasured variable, can be estimated through four different ways. Since there are multiple ways of estimating it, stream 8 can also be regarded as a redundant variable despite the fact that it is unmeasured. The concept of redundancy hitherto used by previous authors is applicable only to measured variables, and is therefore inadequate to characterize this type of redundancy.

2. The concept of redundancy also does not adequately describe the degree of redundancy. In other words, the number of different ways through which a variable can be estimated is not accounted for by the redundancy concept.

In contrast, the concept of reliability that we proposed (Ali and Narasimhan, 1993) does not have these defects since it applies equally to both measured and unmeasured variables. Furthermore, it is clear from its definition that the reliability of a variable increases as the number of ways of estimating

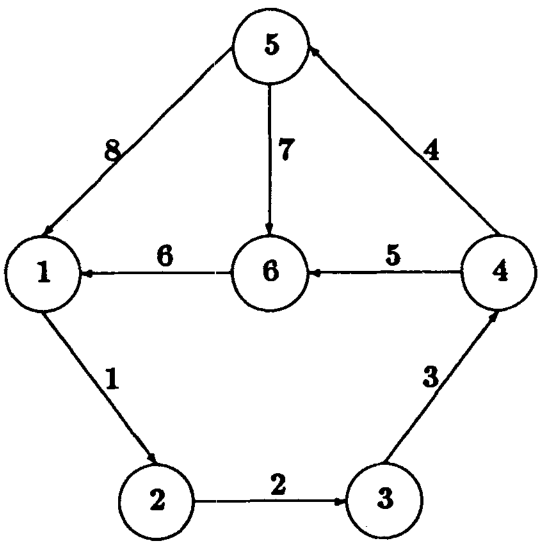

Figure 1. Simplified ammonia network.

the variable. In the following section, we describe how the reliability of variables can be computed.

Reliability Computation

Let the variable i be deleted from each of the cutsets through which it can be indirectly estimated, that is, from the cutset members of Q_i^m. Let the resulting sets be denoted as $A_1, A_2, \ldots, A_m$. Clearly, i can be indirectly estimated if and only if all the sensors measuring the edges of at least one set A_k are working. We denote a set A_k as active if the preceding condition holds. Then the reliability of estimating i indirectly is given by

$$R'(i) = Pr\{A_1 \cup A_2 \cup \ldots \cup A_m\}. \tag{2}$$

Since, there may be edges in common between these sets A_k, the reliability computation is nontrivial.

This type of problem also occurs in reliability computation of communication networks where it is required to compute the reliability with which two stations (nodes) can communicate in the presence of failure of connecting communication links. Several algorithms are available for the evaluation of Eq. 2 (Veeraraghavan and Trivedi, 1991; Fong and Buzacott, 1987; Tewari and Verma, 1980). The most popular of them are based on the so-called sum-of-disjoint products (sdp) formulation of Eq. 2, which can be rewritten as:

Table 1. All Cutsets of Ammonia Network

Sl. No.	Cutset	Sl. No.	Cutset	Sl. No.	Cutset
1	5, 6, 7	7	1, 6, 8	13	1, 4, 5
2	1, 2	8	1, 4, 6, 7	14	2, 4, 6, 7
3	1, 3	9	1, 5, 7, 8	15	3, 4, 6, 7
4	2, 3	10	2, 4, 5	16	3, 4, 5
5	2, 6, 8	11	4, 7, 8	17	2, 5, 7, 8
6	3, 6, 8	12	4, 5, 6, 8	18	3, 5, 7, 8

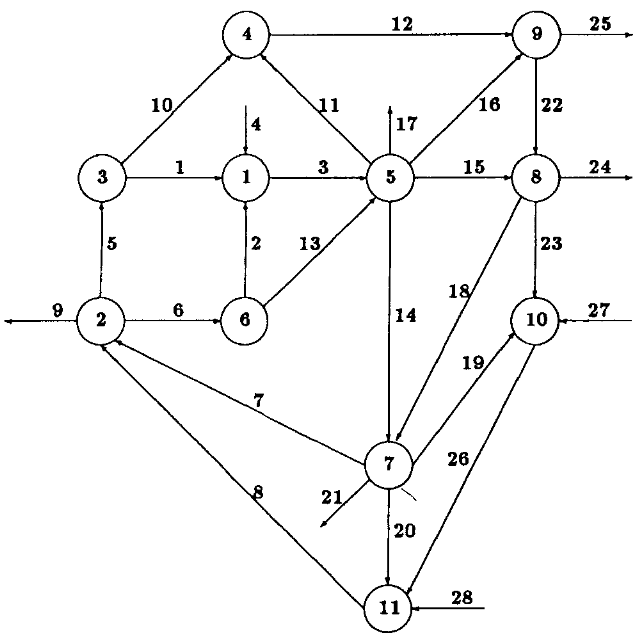

Figure 2. Steam-metering network.

$$R'(i) = Pr\left\{A_1 \cup \bar{A}_1 A_2 \cup \ldots \bar{A}_1 \bar{A}_2 \ldots \bar{A}_{m-1} A_m\right\} \tag{3}$$

$$= Pr\{A_1\} + Pr\{\bar{A}_1 A_2\} + \ldots + Pr\left\{\bar{A}_1 \bar{A}_2 \ldots \bar{A}_{m-1} A_m\right\} \tag{4}$$

where $\bar{A}_k$ denotes that the set A_k is inactive.

The sdp formula gets its name from the fact that the terms in the probability expression are mutually disjoint. Thus, the task is to compute each of the terms in Eq. 4.

We have developed an algorithm for computing the reliabilities based on the sdp formula. The algorithm is described in Appendix B and a more detailed analysis of the algorithm is available in Ali (1993).

A comparison of our algorithm with those of Fong and Buzacott (1987) and Tewari and Verma (1980) shows that it performs more efficiently and requires less memory. For example, all three algorithms were tested on the steam-metering network (Serth and Heenan, 1986) shown in Figure 2, after deleting nodes 10, 11, and 12 from it. Assuming that all edges are measured, we find that stream 1 can be estimated (indirectly) through 61 ways. Table 2 presents the results of applying the three algorithms for evaluating the reliability of variable 1. Column 2 of this table gives the maximum number of subsets of edges that need to be stored during the execution of the algorithms and is an indication of their memory requirements. In column 3, the computing time for executing the algorithms on a PC-386 equipped with a 387 coprocessor is given. The results clearly show that our algorithm requires less computational time and considerably less memory. If we delete only nodes 11 and 12 from this network and measure all the edges, then there are 105 ways of estimating variable 1. Although our algorithm could compute the reliability of this variable in 33 seconds, the other two algorithms could not be executed on the PC due to memory limitations. As shown later, the complete steam-metering network could also be tackled by our algorithm. It should also be noted that the reduction in computation time is significant, since in sensor network design, the preceding algorithm is used repeatedly for computing the reliability of all variables for every sensor network.

Table 2. Performance of Reliability Computation Algorithms for Steam-Metering Network

Algorithm	Max. Sets	Time (s)
Tiwari and Verma	1,472	10.48
Fong and Buzacott	368	6.20
Ali	68	5.67

Equation 2 considers only the indirect ways of estimating a variable. In general, for measured and unmeasured variables the reliability of estimating i is given by:

$$R(i) = \begin{cases} R'(i) & \text{if } i \text{ is unmeasured} \\ 1.0 - [1 - R'(i)]p_i & \text{if } i \text{ is measured} \end{cases} \tag{5}$$

where $R'(i)$ is given by Eq. 2.

Sensor Network Design

In this section, we develop an algorithm for a redundant sensor network design using a *given* number of sensors, n_s, where n_s is greater than $n_{\min}$. The sensor network is designed to maximize the minimum reliability among all variables. For any sensor network design, the minimum reliability among all variables is also defined as the network reliability. We first describe the design procedure assuming all variables are important and are measurable. We also assume that a variable is measured at most using one sensor. Extensions of this procedure to handle specifications on variables and to treat hardware redundancy are subsequently described.

The algorithm that we develop starts with an initial solution and successively improves it. In each iteration, a sensor from some measured variable is removed and placed on some other variable, such that the network reliability improves. The variable from which a sensor is removed is denoted as the entering variable while the new variable being measured is called the leaving variable. In this respect, the procedure is similar to the algorithm SENNET described by Ali and Narasimhan (1993). There are, however, some important differences between the algorithms as described below:

1. In SENNET, the entering and leaving variables were chosen so that the unmeasured variables always formed a spanning-tree structure. For a redundant sensor network, however, the only condition that has to be met for ensuring the observability of all variables is that unmeasured variables do not form a cycle (Mah, 1976). One possibility is to first obtain and store all cycles of the graph, which can be used for checking, when the choices of entering and leaving variables are made in every iteration. Such a computationally cumbersome check, however, is not necessary since our iterative improvement technique automatically ensures observability of all variables at every iteration as described later.

2. SENNET requires only the fundamental cutsets of the process graph with respect to a spanning tree for computing the reliabilities of unmeasured variables. In the present case, however, all cutsets of the network have to be obtained and

used for computing the reliability of variables as described in the previous two sections.

3. Another important difference is that in the minimum number of sensors case, the minimum reliability (network reliability) is always attained by an unmeasured variable, whereas in this case the minimum reliability may be achieved by either a measured or an unmeasured variable. This implies that when we attempt to improve the network reliability, we cannot focus on unmeasured variables only, but have to compute the reliabilities of all variables.

We first discuss the choice of an initial solution followed by the choices of entering and leaving variables.

Initial solution

An initial solution that ensures observability of all variables is generated simply as follows. We obtain a spanning tree of the process graph and pick all its chords and any $n_s - n_{\min}$ branches of the tree to be measured streams. In this initial solution, all unmeasured variables will be observable, since measuring the chords of a spanning tree itself is sufficient to ensure observability of all variables and the addition of more sensors cannot destroy observability.

Choice of leaving variable

Let x be the variable that is estimated with the least reliability in some iteration. A new variable is chosen to be measured such that it improves the reliability of x. We first consider the different ways through which the reliability of variable x can be indirectly improved by selecting some other variable to be measured.

In the reliability evaluation of x, only those cutsets of the graph that contain x and that do not contain any other unmeasured edge are useful. Thus, the leaving variable can increase the reliability of x, only if the following two conditions are satisfied:

1. The leaving variable is a member of some cutset in which x is also a member.

2. The cutset does not contain any other unmeasured edge, besides x and the leaving variable.

These observations lead to the following procedure for selecting the leaving variable. In any iteration, we first obtain the set, Q_x^u, of all cutsets containing x that have only one unmeasured edge, other than x. Let $K_{x,q}^u$ be a cutset member of Q_x^u that contains unmeasured edge q. We can choose q as a leaving variable candidate. This gives at least one additional way of indirectly estimating x, thereby increasing the reliability of x. It should be noted that the number of ways of indirectly estimating x may increase by more than one if there are many $K_{x,q}^u$ in which q is the only unmeasured edge. If there are many candidates for the leaving variable, the one that gives the maximum number of additional ways of indirectly estimating x should be selected first.

These observations hold whether x happens to be a measured or an unmeasured variable. If x is an unmeasured variable, then it can itself be a leaving variable candidate. In our implementation, if x is unmeasured, then it is selected as the first choice of leaving variable. Thus, the set of leaving variable candidates, O, is given by

$$O = \{q | q \in K_{x,q}^u;\ q \text{ is unmeasured}\} \cup \{x | x \text{ is unmeasured}\}. \tag{6}$$

Choice of entering variable

In order to keep the number of sensors fixed, a sensor from a measured variable has to be removed. The removal of a sensor may decrease the reliability of x as well as that of other variables. The entering variable should therefore be chosen such that the network reliability does not decrease. Since there are multiple ways of estimating a variable, it is not possible to predict which sensor removal will meet this condition. We can resort to explicit enumeration where each sensor is removed in turn and compute the reliabilities to check whether the condition is satisfied. The following observations are used to reduce the search space and, hence, decrease the computational burden.

1. Our objective is to improve the reliability of the least reliable variable, x. In fact, the leaving variable was chosen so as to provide at least one additional way for estimating x. If the entering variable is a member of the set $K_{x,q}^u$, then it would nullify this gain. If there is only one cutset $K_{x,q}^u$, then the entering variable cannot be a member of this cutset. However, when there are multiple cutsets $K_{x,q}^u$, then the entering variable cannot be a common member of all such $K_{x,q}^u$. This ensures that there will be at least one additional new way of indirectly estimating variable x in the new sensor network. The set I of entering variable candidates is therefore given by

$$I = N^m - \bigcap_i K_{x,i}^u \tag{7}$$

where N^m is the set of measured edges.

2. As pointed out earlier, if the number of ways of estimating a variable increases, then the reliability of the variable also increases. Furthermore, if the cardinality of a cutset containing variable i is high, then its contribution to the reliability of i *tends* to be small (although, the failure probabilities of sensors on other edges of this cutset and the interaction of this cutset with other cutsets through which i is estimated also have an effect).

Using, the preceding observations a measure is proposed to rank the entering variable candidates, so that the most promising candidates that can improve the network reliability are considered first. Let Q_x^m be the set of cutsets through which x can be indirectly estimated for the current sensor network. For each entering variable candidate i, the following measure is computed.

$$M(i) = \frac{\text{Sum of cardinalities of cutsets of } Q_x^m \text{ and } K_{x,q}^u \text{ in which } i \text{ is a member}}{\text{Number of cutsets of } Q_x^m \text{ and } K_{x,q}^u \text{ in which } i \text{ is a member}}. \tag{8}$$

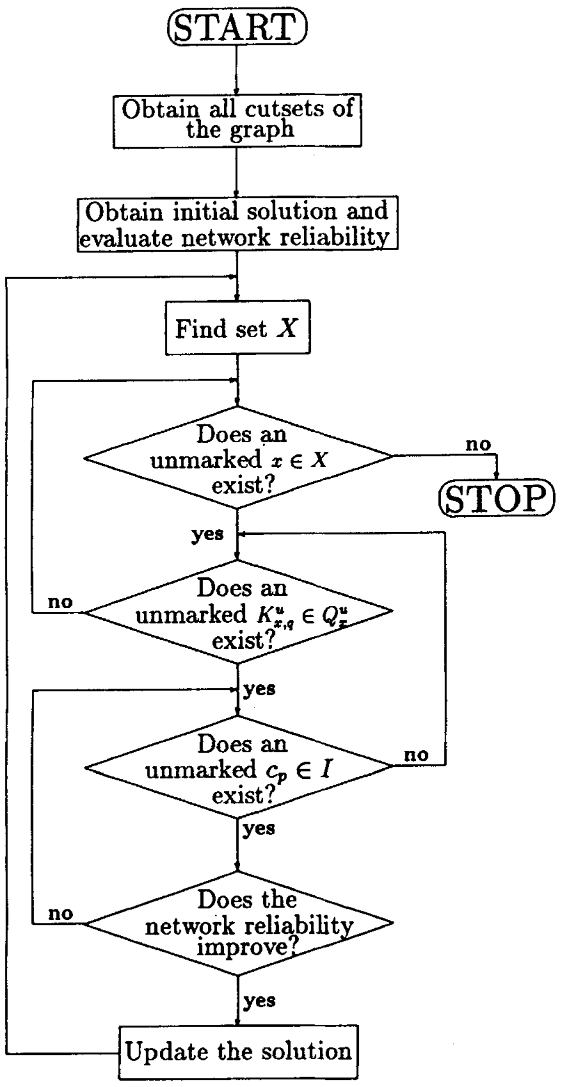
START
Obtain all cutsets of the graph
Obtain initial solution and evaluate network reliability
Find set X
Does an unmarked $x \in X$ exist?
no
STOP
yes
Does an unmarked $K^u_{x,q} \in Q^u_x$ exist?
no
yes
Does an unmarked $c_p \in I$ exist?
no
yes
Does the network reliability improve?
no
yes
Update the solution

Figure 3. Algorithm GSENNET.

If an entering variable candidate is not present in any of the cutsets of Q^m_x or $K^u_{x,q}$, then its measure is set to some high value. The entering variable candidates are ranked in decreasing order of $M(i)$. Note that a high value of $M(i)$ implies that i occurs in fewer cutsets through which x can be estimated in the new sensor network, or, if it occurs, then the cardinality of such cutsets may be large. Thus, if the sensor that measures i is removed, then it may not decrease the reliability of x substantially.

The entering variable candidates are each chosen in turn according to their ranking, and the sensor from it is temporarily deleted. The reliabilities of all variables are computed. If the minimum reliability over all variables is greater than the current network reliability, then the choice holds. Otherwise, we try the next candidate.

Algorithm

Based on the preceding discussion, the steps of the iterative improvement algorithm are given below. For ease of reference we call this Generalized SENsor NETwork design algorithm GSENNET. A flowchart describing the algorithm is also presented in Figure 3 for convenience.

Step 1. Obtain all cutsets of the process graph.

Step 2. Obtain a spanning tree and generate an initial solution by placing sensors on all the chords and remaining sensors on some of the branches of the spanning tree. Compute the network reliability.

Step 3. Find the set of edges X that has the least reliability.

Step 4. Select an unmarked element $x \in X$ and mark it. If no unmarked edge exist, then stop.

Step 5. Obtain the set O of leaving variable candidates (Eq. 6). Arrange the variables in set O in decreasing order based on the number of cutsets of Q^u_x in which they occur.

Step 6. Pick the next unmarked edge $q \in O$ and mark it. If an unmarked element does not exist, go to step 4. Otherwise select q as a leaving variable and go to step 7.

Step 7. Find the set I of entering variable candidates (Eq. 7). Compute their measures (Eq. 8) and arrange them in order of decreasing measures.

Step 8. Choose the next unmarked element $p \in I$ and mark it. If an unmarked p does not exist, go to step 6. Otherwise select p as the entering variable.

Step 9. Evaluate the network reliability. If the network reliability improves, accept p as an entering variable and go to step 3. If the network reliability does not improve, go to step 8.

It should be noted that the algorithm just described may not give globally optimal solutions, because in any iteration we only search for a better *neighboring solution* that differs from the current solution in the placement of one sensor. If a better neighboring solution cannot be obtained, then this is similar to obtaining a locally optimal solution. It should also be noted that in the preceding algorithm there is no need to check whether the solution obtained at any iteration ensures the observability of all variables. This is due to the fact that we start with an initial solution that ensures observability and at each subsequent iteration increases the network reliability. A positive value of network reliability automatically implies that all variables are observable.

Example 2

The following example illustrates the application of GSENNET. Consider the simplified ammonia network shown in Figure 1 in which a minimum of three sensors is required to observe all variables. Suppose that we wish to design a redundant sensor network using four sensors. For reasons of simplicity we assume that all sensors have equal failure probability of 0.1.

Step 1. We evaluate all cutsets of the network. These cutsets are given in Table 1.

Step 2. The initial solution is generated using the spanning tree 1, 2, 3, 7, 8 in which edges 4, 5, and 6 are measured. Since we need to install one more sensor, we arbitrarily select edge 1 from this spanning tree to be also measured. Thus, in the initial solution edges 1, 4, 5, and 6 are measured. Using the reliability algorithm and Eq. 5, we obtain the following reliabilities.

$R(1) = 0.981$; $R(2) = 0.981$; $R(3) = 0.981$; $R(4) = 0.981$
$R(5) = 0.981$; $R(6) = 0.900$; $R(7) = 0.972$; $R(8) = 0.972$.

Step 3. Since edge 6 has the least reliability, $X = \{6\}$. Note that although edge 6 is measured, it has least reliability among all variables.

Steps 4 and 5. X contains only one element, we mark it and go to step 6.

Steps 6 and 7. Using Table 1 we obtain the set Q_6^u, which contains the following four cutsets.

1. $\{5, 6, \underline{7}\}$
2. $\{1, 6, \underline{8}\}$
3. $\{1, 4, 6, \underline{7}\}$
4. $\{4, 5, 6, \underline{8}\}$

In these cutsets, unmeasured edges are indicated by an underscore. The set of leaving variable candidates is given by $O = \{7, 8\}$. Note that both variables occur in two cutsets of Q_6^u. We choose edge 7 as the leaving variable. This gives two more cutsets through which mass flow in edge 6 can be estimated. Note that cutsets $K_{6,7}^u$ are $\{5,6,7\}$ and $\{1,4,6,7\}$.

Step 8. The set of entering variable candidates are $I = \{1,4,5\}$, with measures $M(1) = 4.0$, $M(4) = 4.0$, $M(5) = 3.0$. We select edge 1 as the entering variable. Now the new set of measured edges is $\{4,5,6,7\}$ and the network reliability becomes 0.882.

We note that the reliability of the network has decreased from 0.900 to 0.882, if we select edge 1 as the entering variable. Furthermore, we find that the selection of edge 4 as the entering variable also does not improve the network reliability. When we choose edge 5 as the entering variable, we find that the network reliability improves from 0.900 to 0.964 and therefore the choice of edge 5 as the entering variable holds. We also find that on further iterations the network reliability does not improve. This shows that a local optimal solution has been achieved by the algorithm. Comparing this solution to the one obtained by explicit enumeration we find that this solution is also the global solution.

Extensions of Sensor Network Design Algorithm

In this section, we describe modifications to GSENNET for treating the following two cases:

- Use of redundant sensors for measuring a variable (hardware redundancy).
- Treatment of specifications on variables such as those that are measurable and those that are important.

Treatment of hardware redundancy

The use of two or more sensors for measuring a variable is not uncommon in a plant. In such cases, one of the sensors may be a field instrument and the other a remote one that is generally used for control and other applications. Here, we assume that if a variable is measured using two or more sensors, then all the sensor data are used in estimating the variables. For simplicity, we also assume that at most two sensors are used to measure a variable, although the method we propose can be easily generalized for a higher degree of hardware redundancy.

In order to treat hardware redundancy, the procedure for obtaining the entering and leaving variable candidate sets is modified. The leaving variable candidate set given by Eq. 6 includes only unmeasured variables. In order to allow for hardware redundancy, all measured variables that occur in cutsets K_x^m and that are currently measured by only one sensor can also be leaving variable candidates, since these can also increase the reliability of variable x. In the case of entering variables, the candidate set given by Eq. 7 can now also include all measured variables that are being currently measured by more than one sensor. Both these modifications increase the search space, leading to a corresponding increase in computational requirements. Since, it has been observed that hardware redundancy does not increase the reliability as much as spatial redundancy, the first choice of leaving variable can be an unmeasured variable. Similarly, the first choice of an entering variable is one that is measured by more than one sensor.

In addition to the preceding modifications of leaving and entering variable candidate sets, the computation of reliabilities has to be modified slightly to account for variables that are measured by two sensors. We note that unless both the sensors measuring a particular variable fail, the observability of variables is not affected. If p_1 and p_2 are the failure probabilities of two sensors measuring a variable, then the probability that both of them fail is $p_1 \times p_2$, assuming that the sensors fail independently. Thus, for the purpose of computing the reliabilities, the two sensors on a particular variable can be treated as equivalent to one sensor with failure probability $p_1 \times p_2$.

Treatment of specifications on variables

In a process some variables may not be important and therefore need not be observable. Similarly, some variables may be unmeasurable due to technical and other economic reasons. Vaclavek and Loucka (1976) and later Madron and Veverka (1992) have treated such specifications in their measurement placement strategies. In this section, we describe modifications to our sensor network design algorithm for handling such specifications on variables. We define the following set of variables based on the specifications.

1. U: Set of unmeasurable variables.
2. $U^r \subseteq U$: Set of important but unmeasurable variables.
3. $U^{nr} \subseteq U$: Set of unimportant and unmeasurable variables.
4. V: Set of measurable variables.
5. $V^r \subseteq V$: Set of measurable and important variables.
6. $V^{nr} \subseteq V$: Set of measurable but unimportant variables.

In the design of a sensor network only the reliabilities of important variables (whether measurable or not) are of concern. As before, we attempt to maximize the minimum reliability among all *important* variables. Due to the preceding specifications, the problem may become ill-posed in the sense that there may not be any sensor network design that can ensure observability of all important variables. We first check the following conditions to see whether the problem is well-posed.

1. The sets U^r, U^{nr}, V^r, and V^{nr} should be mutually disjoint. Moreover, $U \cup V$ should be the entire set of variables.
2. The subgraph formed by the variables (streams) in U

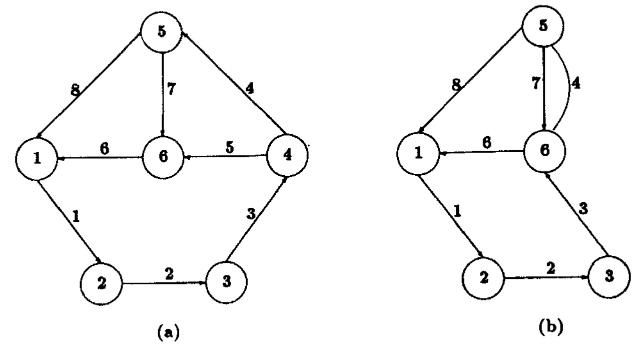

Figure 4. Node-merging operation.

should not contain any cycle in which one or more variables of U^r are present.

The first of these conditions follows from the definition of the sets given earlier. If the second condition is violated, then any sensor network design will contain a cycle of unmeasured variables. One or more variables in this cycle will belong to U^r and these important variables will be unobservable (Mah, 1976).

If the conditions are satisfied, we also need to check whether the given number of sensors n_s is sufficient to ensure observability of all important variables. The minimum number of sensors required in this case is no longer given by Eq. 1, but is computed using the procedure described below. The procedure we describe not only gives the minimum number of sensors, but also reduces the size of the process graph for the purpose of sensor network design.

Minimum number of sensors

First, we outline the process of merging two nodes in the network. A pair of nodes a and b in a graph is said to be merged if the two nodes are replaced by a single node, such that every edge that was incident on either a or b (but not on both) is incident on the new node. The edges that were incident on both a and b, called internal edges, disappear from the new graph. This is illustrated by the following example. Consider the ammonia network redrawn in Figure 4a. The nodes 4 and 6 are merged to form a new node 6′. Edges 3 and 4, which were incident on node 4, and edges 6 and 7, which were incident on node 6, are now incident on node 6′ shown in Figure 4b. Edge 5, which was incident on both nodes 4 and 6, is eliminated and is not present in Figure 4b.

In the graph reduction process, we merge every pair of adjacent nodes on which any edge from set U^{nr} is incident. If any other edge is eliminated in this process, then it implies that this edge along with some edges of set U^{nr} forms a cycle of the original graph. Thus, we can immediately infer from condition 2 that no edge from set U^r can be eliminated, although edges from set V^{nr} and/or from set V^r may be eliminated during the node-merging process. If an edge from set V^{nr} is eliminated, then it implies that such an edge need not be measured at all. This is due to the fact that any cutset of the original process graph containing this variable will also contain at least one edge from set U^{nr} and, thus, this variable will not be useful for indirectly estimating any important variable. On the other hand, if an edge from set V^r is eliminated, then this edge should always be measured. The reason for this is that any cutset of the original process graph that contains this variable will also contain at least one edge from the set U^{nr}. This implies that there is no way of indirectly estimating this important variable, and therefore it must be measured.

After all the edges in set U^{nr} have thus been eliminated, we obtain a reduced process graph. Let e_d be the set of edges from set V^r that are eliminated. After obtaining the reduced graph, we still have to check whether variables from set V^{nr} need to be measured. A variable from set V^{nr} need not be measured unless it is definitely required to estimate at least one of the important variables. For identifying such variables, we start constructing a spanning tree of the reduced network by first choosing all edges from set U^r as part of the tree. It should be noted that this is possible because edges of set U^r do not form cycles. We then attempt to add one edge at a time from set V^{nr} to the tree, provided it does not form a cycle with other edges of the tree. If at any step, an edge from set V^{nr} forms a cycle containing only other edges of set V^{nr} that are in the tree, then this edge is marked. After all edges from set V^{nr} have been considered, the spanning tree can be completed by using edges from set V^r. If e_{nr} is the number of marked edges of V^{nr}, then the number of sensors required to observe all important variables is given by

$$n_{\min} = e' - n' + 1 + e_d - e_{nr} \tag{9}$$

where e' and n' are the number of edges and nodes, respectively, of the reduced graph.

The preceding result simply follows from the fact that for estimating all variables in the reduced process graph, the minimum number of sensors is equal to $e' - n' + 1$. Moreover, e_d edges of set V^r that were eliminated should be measured and e_{nr} marked edges from set V^{nr} present in the reduced graph need not be measured since they are not useful for estimating any of the important variables.

The method also provides an opportunity to reduce the computation required for sensor network design for any given number of sensors, n_s, greater than or equal to $n_{\min}$. The sensor network design algorithm can be applied to the *reduced process graph* using $n_{\min} - e_d$ sensors to obtain their optimal locations. The remaining e_d sensors are placed on edges belonging to set V^r that were eliminated from the original process graph. Moreover, an initial sensor network design solution can be obtained from the spanning tree of the reduced graph using the procedure just described.

The reduced graph still contains all variables from set U^r, and some of the variables from set V^r and V^{nr}. Therefore, minor modifications are made to the sensor network design algorithm for treating such variables. First, the reliabilities of important variables only are computed in order to obtain the network reliability at any iteration. Second, all variables belonging to set U^r are always eliminated from leaving variable candidate set O, at each iteration. This ensures that in the final solution no sensors are placed on such variables.

Treatment of other criteria for sensor network design

In this article, as well as the previous one (Ali and Narasimhan, 1993), we have designed a sensor network that

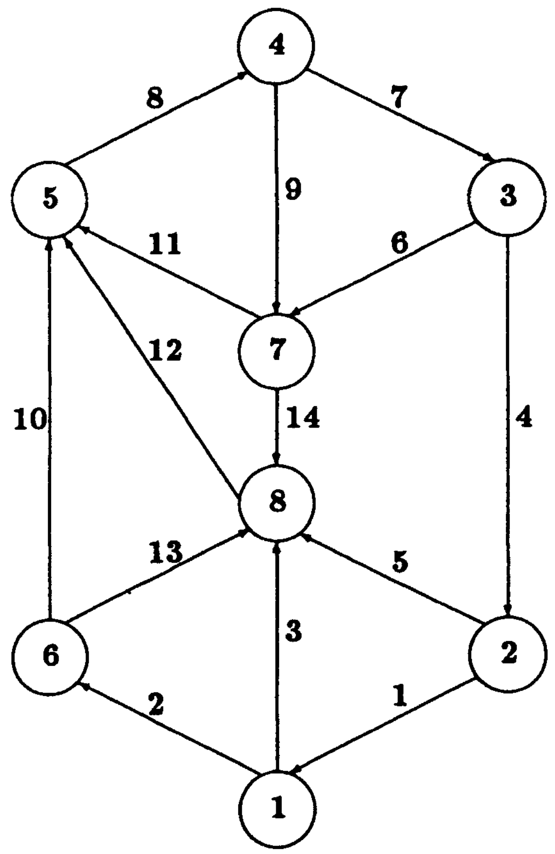

Figure 5. HDA process network.

maximizes network reliability. The basic methodology used in SENNET and GSENNET can also be used, in principle, for designing a sensor network to satisfy other criteria *as long as there is a procedure for evaluating the criteria for a given sensor network design.* For example, we briefly indicate how SENNET or GSENNET can be used for maximizing accuracy of estimation of variables. A comprehensive development and testing of the approach using this criterion is beyond the scope of this article.

A measure of estimation accuracy for a given sensor network has been defined and used by Kretsovalis and Mah (1987a). In order to use such a measure it is necessary to ensure the observability of all important variables. Thus, an initial solution can be obtained, from a spanning tree just as in SENNET or GSENNET. At each iteration the entering and leaving variable choices can be made such that estimation accuracy increases. Kretsovalis and Mah (1987a) have developed explicit analytical expressions for computing the increase (or decrease) in estimation accuracy due to the addition (or removal) of a sensor. These can be used to judge the change in estimation accuracy at each iteration of SENNET or GSENNET. Since the structure of the solution is exploited in these algorithms, they can lead to a more efficient search strategy than any arbitrary combinatorial search.

Results and Discussion

The sensor network design algorithm that we have developed in the previous two sections is applied to two examples, to tests its efficiency.

Table 3. Data and Results of HDA Process*

Data
No. of nodes: 8
No. of edges: 14
Sensor failure probability: 0.1

Results
Minimum no. of sensors: 7

No. of Sensors	No. of Feas. Sol.	No. of Opt. Sols.	Global Reliab.	Optimal Solution (Algorithm)	Opt. Rel. (Algorithm)
7	992	55	0.729	1 5 7 9 12 13 14	0.729
8	1,779	1	0.889	2 5 6 8 10 14	0.881
9	1,614	1	0.955	2 5 7 8 10	0.953
10	929	1	0.975	1 2 7 8	0.975
11	358	1	0.986	1 6 10	0.986
12	91	1	0.989	1 6	0.989
13	13	1	0.997	4	0.997
14	1	1	0.998	—	0.998

*Equal sensor failure probabilities.

Example 3

As a first example, we consider the simplified process of hydrodealkylation (HDA) of toluene (Douglas, 1988) shown in Figure 5.

We assume that all variables are important and measurable. From Eq. 1, we find that at least seven sensors are required for observing all variables. The results of the sensor network design for a given number of sensors starting with 7 up to a maximum of 14 are shown in Tables 3 and 4. The results of Table 3 are obtained for the case of equal sensor failure probabilities, and those of Table 4, for the case of unequal sensor failure probabilities. To compare the results obtained by our algorithm, we have also reported the optimal design obtained by explicit enumeration of all solutions. In these tables, each row gives the sensor network design results

Table 4. Data and Results of HDA Process*

Data
No. of nodes: 8
No. of edges: 14
Sensor failure probability: 0.32 0.16 0.25 0.37 0.09 0.23 0.17 0.10 0.14 0.27 0.34 0.24 0.28 0.12

Results
Minimum no. of sensors: 7

No. of Sensors	No. of Feas. Sol.	No. of Opt. Sols.	Global Reliab.	Optimal Solution (Algorithm)	Opt. Rel. (Algorithm)
7	992	2	0.433	3 4 5 6 12 13 14	0.433
8	1,779	1	0.693	1 3 4 6 10 11	0.693
9	1,614	1	0.837	1 4 6 11 13	0.833
10	929	1	0.897	1 4 6 11	0.881
11	358	1	0.945	1 4 6	0.945
12	91	1	0.960	4 6	0.960
13	13	1	0.967	4	0.967
14	1	1	0.967	–	0.967

*Unequal sensor failure probabilities.

for a specified number of sensors. The number of sensors used are shown in the first column. In column 2, the number of feasible sensor networks that ensure the observability of all variables is given. Column 3 gives the number of optimal solutions (corresponding to maximum network reliability), and column 4 gives the maximum network reliability. The results of columns 2, 3, and 4 are obtained by explicit enumeration of all solutions, and computing their network reliabilities. In column 5, the unmeasured variables in the best solutions obtained by applying GSENNET are shown, and the last column gives the network reliability of the corresponding solution.

For obtaining the results of Table 3, we have assumed all sensor failure probabilities to be equal to 0.1. From columns 2 and 3, we observe that less than 1% of the feasible solutions are optimal. In fact, there is only one optimal solution when more than a minimum number of sensors are used. Nevertheless, our sensor network design algorithm is able to find the optimal solution in most cases. When eight or nine sensors are used, our algorithm is not able to obtain the exact optimal solution, possibly because the number of feasible solutions is large in these cases. Nevertheless, the solution obtained by our algorithm differs from the optimal by less than 0.5 percent. The results of Table 4 are obtained for unequal sensor failure probabilities as shown, where the sensor failure probabilities have been generated using uniform random numbers in the interval 0.05 to 0.4. From the results of this case, similar inferences as before can be made. The results also show that for this process an appropriate choice would be to use 11 sensors, since the network reliability increases only marginally for every additional sensor above 11.

Example 4

We now illustrate the performance of the algorithm using the steam distribution network (Serth and Heenan, 1986) shown in Figure 2. The failure probabilities of sensors are assumed to be all equal to 0.1. The results for this network are given in Table 5, for the number of sensors ranging from 17 to 22. Column 1 of the table gives the number of sensors installed. The best solution obtained by the algorithm is given in column 2 of the table. The corresponding network reliability and the least reliability variables are listed in columns 3 and 4, respectively. For this network, it is not practical to enumerate all solutions explicitly. Therefore, the optimality of solutions given by our algorithm is not verified for this example problem. It is encouraging to observe, however, that the reliability of the sensor network design obtained by our algorithm is increasing as the number of sensors increases. The network reliability increases substantially when one or two sensors more than the minimum are added, but increases more slowly for subsequent additional sensors. This behavior is similar to that obtained for the HDA process example. Thus, we can reasonably conclude that the results obtained are acceptable.

Table 5. Data and Results of Steam Metering Network*

Data			
No. of nodes: 12			
No. of edges: 28			
Failure probability for each sensor: 0.1			
Results			
Minimum no. of sensors: 17			
No. of Sensors	Optimal Solution (Algorithm)	Net. Rel.	Least Rel. Variable
17	2 4 9 10 12 17 20 24 25 26 28	0.531	17, 27
18	1 2 4 7 12 14 16 24 27 28	0.703	14
19	1 2 7 10 12 14 22 27 28	0.819	14
20	1 2 7 10 12 17 23 26	0.859	17
21	1 2 7 10 12 17 26	0.882	17
22	1 2 7 10 12 26	0.930	1

*Equal sensor failure probabilities.

The previous two examples were solved on a PC 386 computer equipped with a 387 coprocessor. While the HDA process example required less than a minute to solve, the computation time for the steam-metering process ranged from about 1.5 h for 17 sensors to about 6 h for 22 sensors. More than 98% of the time was spent on evaluating reliabilities of variables. Thus, the reliability computation algorithm that we have developed plays a vital role in reducing the computation time. There is a need, however, to further improve the efficiency of reliability computation in order to solve larger problems. For example, an attempt to solve the steam-metering example using 27 sensors showed that about 10 h are required to compute the reliabilities of all variables for a given sensor network design, indicating that a few days would be required for obtaining the optimum sensor network design.

Finally, using the steam distribution network as an example, we present results when constraints are imposed on variables. We impose the specifications that mass flows of streams 3, 13, 17 are unmeasurable but important (U^r), mass flows of streams 11, 16, 19, 23 are unmeasurable and unimportant (U^{nr}), and mass flows of streams 2, 12, 18 are measurable and unimportant (V^{nr}). The remaining streams are measurable and important. After elimination of all edges of set U^{nr} by nodal aggregation, the reduced graph shown in Figure 6, is obtained. In this process, edges 12 and 18 from set V^{nr} are also eliminated, but none of the edges from set V^r is eliminated ($e_d = 0$). The reduced graph therefore contains 8 nodes and 22 edges, as shown in Figure 6. Since edges of V^{nr} do not form a cycle in the reduced graph, e_{nr} is equal to zero. Therefore, the minimum number of sensors required to observe all important variables is $22-8+1+0-0=15$ (Eq. 9). The results for this reduced network are given in Table 6. The number of sensors installed in the network are given in column 1 and the specifications on the constraints are listed in column 2 of the table. The best solution obtained by the algorithm, the corresponding network reliability, and the variables having the least reliability are given in columns 3, 4, and 5, respectively. Since explicit enumeration of all solutions is not practical, the trend of results is analyzed for consistency. It is observed from the first three rows of Table 6 that the network reliability increases with an increase in the number of sensors, which is consistent with our expectation.

More interesting consistency checks are obtained by changing the specifications on variables. The results obtained for 17 sensors under different specifications are reported in rows 3 to 7 of Table 6. In order to interpret the results, we first note that by specifying some of the variables as unimportant we are actually relaxing the problem, since the minimum reliability among a subset of variables has to be maximized now,

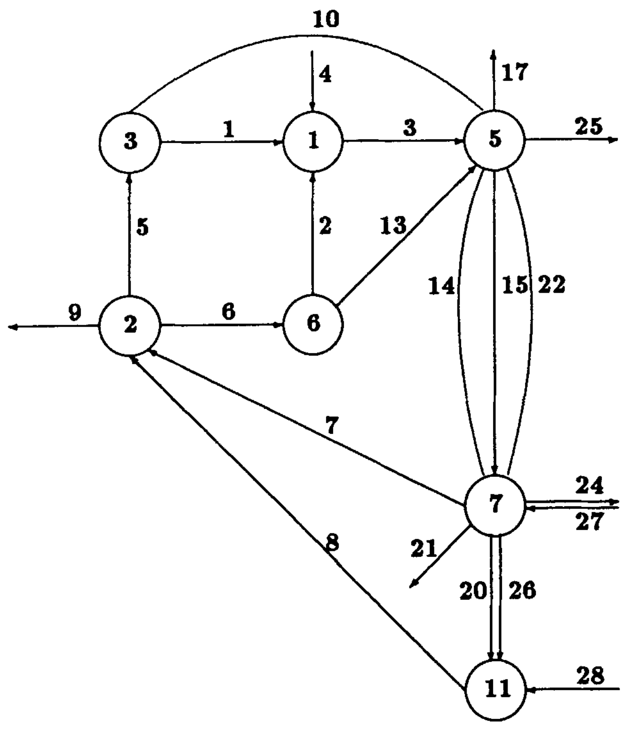

Figure 6. Reduced steam-metering network.

rather than the entire set. Thus, the network reliability for this case must be greater than or equal to that attained for the same process without any specifications. If we specify some of the variables as unmeasurable, however, then the problem becomes more constrained. Therefore, the network reliability in this case must be less than or equal to that obtained under no specifications. Row 4 of Table 6 shows that the network reliability is 0.53 when no specifications are made. The network reliability obtained for the more constrained problem shown in row 5 is lower, while the network reliability obtained for the more relaxed problem shown in row 6 is higher, as expected. The last row shown in Table 6 also corresponds to a relaxed problem. The network reliability, however, is the same as that obtained for the case without specifications (row 4). This is due to the fact that there are two minimum reliability variables, 17 and 27, when no specifications are imposed, as observed from row 4. Out of these only variable 17 is relaxed by specifying it as unimportant for the case presented in the last row, resulting in the same network reliability. Thus, we observe that in all the cases we have tried, our network design algorithm produces results consistent with expectations, confirming its robustness.

Concluding Remarks

In this article, we have developed a generalized graph-theoretic algorithm, GSENNET, for sensor network design using more than the minimum number of sensors. Both hardware redundancy and spatial redundancy are tackled by our algorithm. A procedure has been developed to handle specifications of importance and measurability on process variables. The minimum number of sensors required to ensure observability of all important variables under different specifications has also been computed explicitly. We have shown that the algorithm performs efficiently and robustly on reasonably large practical process examples. Finally, we have indicated

Table 6. Results of Steam Metering Network*

Data
No. of nodes: 12
No. of edges: 28
Failure probability for each sensor: 0.1

Results

No. of Sensors	Constraints/Relaxations	Optimal Solution (Algorithm)	Optimal Rel.	Least Rel. Variable
15	$U^r = \{3, 13, 17\}$ $U^{nr} = \{11, 16, 19, 23\}$ $V^{nr} = \{2, 12, 18\}$	3 5 10 11 12 13 14 16 17 18 19 20 23	0.430	17
16	*ibidum*	3 10 11 12 13 14 16 17 18 19 20 23	0.504	17
17	*ibidum*	3 11 12 13 14 16 17 18 19 20 23	0.511	17
17	$U^r = \varnothing$ $U^{nr} = \varnothing$ $V^{nr} = \varnothing$	1 2 4 9 12 17 21 23 25 27 28	0.530	17, 27
17	$U^r = \{3, 13, 17, 11, 16, 19, 23\}$ $U^{nr} = \varnothing$ $V^{nr} = \varnothing$	3 5 10 11 13 14 16 17 19 20 23	0.430	17, 27
17	$U^r = \varnothing$ $U^{nr} = \varnothing$ $V^{nr} = \{3, 17, 27\}$	1 2 4 9 12 17 21 23 25 27 28	0.590	4, 21
17	$U^r = \varnothing$ $U^{nr} = \varnothing$ $V^{nr} = \{3, 13, 17, 23\}$	1 2 4 9 12 17 21 23 25 27 28	0.530	27

*Specifications on variables.

how the basic methodology used for sensor network design in SENNET and GSENNET can be applied for treating other criteria such as accuracy of estimation.

It may be possible to develop a sensor network design algorithm using matrix methods such as, for example, those adopted by Madron and Veverka (1992). These are likely to be computationally more demanding, however, since they do not in general exploit the structure of the sensor network corresponding to feasible solutions in their search strategy. The extensions of the sensor network design algorithm for tackling nonlinear processes and for satisfying other important objectives needs further research.

Notation

A_i	= set whose sensors are all working
C^i	= set of events
C_j^i	= element j of set C^i
G	= graph
K	= cutset of graph G
K_i	= cutset containing edge i
K_i^u	= cutset containing i and at least one unmeasured edge
$K_{\max}$	= set of maximum cardinality cutsets
m	= number of indirect ways of estimating a variable
p	= entering variable candidate
$R(i)$	= reliability of variable i
x_i	= literal denoting sensor i is active

Literature Cited

Ali, Y., "Sensor Network Design for Maximizing Reliability of Processes," PhD thesis, Indian Inst. of Technol., Kanpur, India (1993).

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

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

Deo, N., *Graph Theory with Applications to Engineering and Computer Science*, Prentice-Hall, Englewood Cliffs, NJ (1974).

Douglas, J. M., *Conceptual Design of Chemical Processes*, McGraw-Hill, Singapore (1988).

Fong, C. C., and J. A. Buzacott, "An Algorithm for Symbolic Reliability Computation with Pathsets and Cutsets," *IEEE Trans. Rel.*, **R-36**, 34 (1987).

Kretsovalis, A., and R. S. H. Mah, "Effect of Redundancy on Estimation Accuracy in Process Data Reconciliation," *Chem. Eng. Sci.*, **42**, 2115 (1987a).

Kretsovalis, A., and R. S. H., Mah, "Observability and Redundancy Classification in Multicomponent Process Networks," *AIChE J.*, **33**, 910 (1987b).

Kretsovalis, A., and R. S. H. Mah. "Observability and Redundancy Classification in Generalized Process Networks: I. Theorems," *Comput. Chem. Eng.*, **12**, 671 (1988a).

Kretsovalis, A., and R. S. H., Mah, "Observability and Redundancy Classification in Generalized Process Networks: II. Algorithms," *Comput. Chem. Eng.*, **12**, 688 (1988b).

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

Mah, R. S. H., *Chemical Process Structures and Information Flows*, Butterworths, Stoneham, England (1990).

Serth, R. W., and W. A. Heenan, "Gross Error Detection and Data Reconciliation in Steam-Metering Systems," *AIChE J.*, **32**, 733 (1986).

Tewari, R. K., and M. Verma, "An Algebraic Technique for Reliability Evaluation," *IEEE Trans. Rel.*, **R-29**, 311 (1980).

Tsukiyama, S., I. Shirakawa, and H. Ozaki, "An Algorithm to Enumerate All Cutsets of a Graph in Linear Time per Cutset," *J. Assoc. Comput. Mach.*, **27**, 619 (1980).

Vaclavek, V., and M. Loucka, "Selection of Measurements Necessary to Achieve Multicomponent Mass Balances in Chemical Plant," *Chem. Eng. Sci.*, **31**, 1199 (1976).

Veeraraghavan, M., and K. S. Trivedi, "An Improved Algorithm for the Symbolic Reliability Analysis of Networks," *IEEE Trans. Rel.*, **R-40**, 347 (1991).

Appendix A: Definitions of Graph Theory Concepts

Spanning Tree of a Graph. A connected subgraph of the graph, which does not contain any cycles and which includes all vertices of the graph.

Branch of a Spanning Tree. An edge of a graph that is present in the spanning tree.

Chord of a Spanning Tree. An edge of a graph that is not present in the spanning tree.

Cutset of a Graph. A minimum set of edges of a graph whose removal disconnects it.

Fundamental Cutsets with Reference to a Spanning Tree. A cutset of the graph that contains only one branch of that spanning tree and zero or more chords of the spanning tree.

Ring Sum of Two Sets. Elements that belong to one or the other set but not to both of them.

Appendix B: Algorithm for Reliability Evaluation

In this Appendix, we present an efficient algorithm for reliability evaluation given the indirect ways $A_1, A_2, \ldots, A_m$ of estimating a variable. A more detailed development of this algorithm can be found in Ali (1993).

Step 0. Let the sets $A_1, A_2, \ldots, A_m$ be arranged in order of increasing cardinality.

Step 1. Evaluate $Pr(A_1)$. That is:

$$r = \prod_{i \in A_1} p_i.$$

Initialize C^1 to single element literal sets of A_1. Set $k = 1$.

Step 2. For every element C_j^k of C^k check if $C_j^k \cap A_l \neq \emptyset$ for all $l \geq k+1$. If so remove C_j^k from C^k.

Step 3. Evaluate $Pr(\bar{A}_1 \bar{A}_2 \ldots \bar{A}_k A_{k+1})$ as follows: Initialize set $M = \emptyset$. Let

$$r^* = \prod_{i \in A_{k+1}} p_i.$$

Step 3*a*. Select any unmarked element (set of literals) of C^k, say C_j^k. If no such element exists go to step 4.

Step 3*b*. If $C_j^k \cap A_{k+1} \neq \emptyset$, then mark C_j^k as unused and return to step 3a; else mark C_j^k as used. Update r and set M as:

$$r = r + r^* \left(\prod_{i \in C_j^k} (1 - p_i) \prod_{\substack{i \in M \\ i \notin C_j^k}} p_i \right)$$

$$M = M \cup C_j^k.$$

Go to step 3a.

Step 4. If $k = m - 1$ stop. Else construct the set C^{k+1} from C^k as follows.

Step 4*a*. All unused elements of C^k are also elements of C^{k+1}.

Step 4*b*. Take each used element C_j^k of C^k in turn and construct the unions $C_j^k \cup x_i$ for every $x_i \in A_{k+1}$. If $C_j^k \cup x_i$ is a superset of any element of C^{k+1} then discard it else add it to C^{k+1}. Increment k and go to step 2.

In the algorithm, the set C^k is one whose elements are sets of literals drawn from the sets A_1, A_2, ..., A_k. The set C^k can be interpreted as follows. If all the literals in a literal set element of C^k fails, then all sets A_1, A_2, ..., A_k will fail. This set is useful in evaluating the $k+1$-th term of Eq. 4 without explicitly generating the mutually exclusive events.

Manuscript received Sept. 20, 1993, and revision received May 23, 1994.