Design of Nonlinear Sensor Networks for Process Plants

DuyQuang Nguyen and Miguel J. Bagajewicz*

University of Oklahoma, 100 East Boyd Street, T-335, Norman, Oklahoma 73019

In this article we extend and generalize the ideas of two previous articles devoted to linear sensor networks to nonlinear systems. In those previous articles the use of cutsets and a decomposition procedure were proposed and proved efficient to solve large scale linear problems. In this article we show that a similar procedure, now based on a variable elimination scheme, can be also used efficiently for medium size nonlinear problems, but its computational efficiency for realistic large scale problems is not satisfactory. We also propose an alternative technique for the case of tree enumeration using instruments instead of equations that is very efficient for heavily instrumented flowsheets.

1. Introduction

The importance of data treatment techniques like data reconciliation and gross error detection has long been recognized. Those data treatment techniques have been the subject of research over the past four decades, and many successful industrial applications of data reconciliation have been reported. Considering the fact that, in reality, only a fraction of process variables are measured by sensors, the precision of the estimators depends on the location of the sensors and the precision of the sensors themselves. Hence the problem of systematically locating sensors that meet prespecified criteria arises naturally, formally known as the sensor network design/retrofit problem.

After the seminal work of Vaclavek and Loucka[1] several papers were published: Kretsovalis and Mah[2] used combinatorial search to minimize cost. Madron and Veverka[3] used Gauss Jordan elimination to achieve observability of key variables at minimum sensor cost. Departing from process monitoring criteria, Ali and Narasimhan[4] introduced the concept of system reliability and proposed a method that maximizes system reliability. Raghuraj et al.[5] and Bhushan and Rengaswamy[6–8] presented sensor network design formulation based on fault diagnosis criteria. Musulin et al.[9] used genetic algorithm in the design of sensor network for principal components analysis monitoring. Bagajewicz et al.[10] designed sensor network for simultaneous process monitoring and fault detection/resolution purpose using a MILP formulation. Meyer et al.[11] used a graph oriented approach for cost-optimal sensor network design with a requirement on observability of key variables. Luong et al.[12] considered several requirements in the design of sensor network, observability of variables required for process control, required degrees of redundancy for some variables, and reliability of the measurement system and minimum sensor cost; the computational method is based on the analysis of cycles of process graph.

Bagajewicz[13] was the first to formulate the sensor network problem as a mixed-integer programming model (in that work, the model is MINLP) using binary variables to indicate whether a variable/stream is measured or not and sought to obtain minimum sensor cost. He also introduced new concepts regarding performance specifications: residual precision, gross errors resilience, and error detectability. In searching for the optimum solution, a tree enumeration algorithm was used, which guarantees a global optimum solution, but its computation requirement inhibits its use in large scale problems. A generalized model for grass root sensor network design and instrumentation upgrade as well as resource allocation was presented by Bagajewicz and Sanchez.[14] Finally, all literature review up to the year 2000 can be found in the book by Bagajewicz.[15]

Chmielewski et al.[16] showed that an unmeasured variable can be modeled in data reconciliation formulation using a fake sensor with very high variance. They used a linear matrix inequalities (LMI) technique to obtain the solution; the technique is applicable to both a linear steady-state system and a linear dynamic system. The appealing point of this approach is that the LMI can be in principle relaxed, and therefore they are amenable to classical branch and bound approaches. However, LMI-based methods can only deal with precision and residual precision constraints. To this date, there is not an LMI version of resilience and gross error detectability constraints. The idea of using a fake sensor with very high variance for an unmeasured variable enabled one to state the performance constraints explicitly in the analytical form and was used by Bagajewicz and Cabrera,[17] who presented a MILP sensor network design formulation. The procedure suffers from scaling problems. Recently, multiobjective sensor network design became attractive, and many researches have been reported. Bagajewicz and Cabrera[18] addressed multiobjective sensor network design using pareto optimal solutions visualization techniques. Sen et al.[19] and Carnero et al.[20,21] used genetic algorithms. Heyen et al.[22] also used a genetic algorithm to design a cost-optimal sensor network that renders required precision of key variables; the computation algorithm can be applied to nonlinear systems by the linearization of process constraints at the nominal operating conditions, assuming steady state. Singh and Hahn[23,24] located sensors for state and parameter estimation of stable nonlinear systems. Muske and Georgakis[25] discussed the tradeoff between measurement cost and process information that is used for control purposes and formulated a pareto optimization problem for finding solutions.

Most recently, Gala and Bagajewicz[26,27] presented an alternative tree enumeration method where at each node combinations of flowsheet cutsets are used. This method has proven to be remarkably faster, especially after a decomposition technique is used.

Most of the aforementioned works were applied to linear systems, that is, when flowrates between units in process plants are to be estimated. Few researchers have published work on sensor network design for nonlinear processes, that is, when the underlying process model includes other balances like component and energy balances as well as other features like VLE relations or reactions in reactors. One such effort was

* To whom correspondence should be addressed. Tel.: 1-405-325-5458. E-mail: bagajewicz@ou.edu.

10.1021/ie0708778 CCC: $40.75 © 2008 American Chemical Society
Published on Web 07/04/2008

presented by Ali and Narasimhan[28] who used a graph oriented method to design a sensor network specifically for bilinear systems but maximizing system reliability instead of cost. Bhushan and Rengaswamy[7,8] presented a general framework applicable to nonlinear systems based on graph theory for finding a reliable sensor network from a fault diagnosis perspective. Single constraint of required precision is considered, and a genetic algorithm was used by Heyen et al.[22] in the design of sensor network for general systems including nonlinear systems. Only the work of Heyen et al.[22] is closely related to our work because of the same perspective/objective, which is designing a cost-optimal sensor network for process monitoring purposes, but their GA-based procedure does not guarantee optimality, neither local nor global. Thus, the design of a nonlinear sensor network from the process monitoring perspective with requirements on observability and redundancy has not yet been successfully tackled and is the objective of this work. Aside from developing the details on how to handle nonlinear systems we focus on an efficient computational method.

In this paper, a branch and bound procedure similar to the one presented by Gala and Bagajewicz[26,27] is used. The algorithm is equation-based rather than cutset-based for reasons that are explained below. First, we discuss the background theory and summarize the difficulties of applying cutset theory to graphs representing nonlinear systems. Next, we present our solution procedures, followed by three illustrated examples. Then we present with examples a specific strategy tailored for highly specified design cases. Finally, the criterion to choose the appropriate strategy is presented.

2. Background

The optimization model to design minimum cost sensor network as presented by Bagajewicz[13] is (in its simplest form) as follows:

$$\left.\begin{array}{c} \operatorname{Min}\sum_{\forall i} c_i q_i \\ \text{s.t.} \\ \sigma_i(q) \le \sigma_i^* \quad \forall i \in M_s \\ q_i = 0, 1 \qquad \forall i \end{array}\right\} \tag{1}$$

where q_i, an element of vector q, is a binary variable indicating that a sensor is used to measure variable i, c_i is the cost of such a sensor, and M_s represents the set of variables where a performance specification is required (variables of interest or "key" variables). This performance specification can be precision of the estimators. Thus, $\sigma_i(q)$ is the precision of estimator i corresponding to the set of sensors represented by q and σ_i* the corresponding threshold value. Other specifications include residual precision, error detectability, and gross error resilience. Realizing that there is no general explicit analytical expression for $\sigma_i(q)$, Bagajewicz[13] proposed a tree enumeration algorithm using the vector q as a basis (thus enumerating combinations of individual sensors), which has the property that every branch node of a feasible node is also feasible and has larger cost, thus allowing some pruning properties. This algorithm guarantees optimum solution,; however, for large scale problems, the computation requirement is so intensive that the use of this algorithm becomes impractical. While the method of transforming the problem into a LMI-based convex MINLP[16] allows the use of the classical branch and bound methods, this can only be applied to precision constraints but not others.

Gala and Bagajewicz[26] realized that instead of exploring single measurements, specific (and meaningful) subsets could be used. These subsets are called cutsets (taken from graph theory), and as Kretsovalis and Mah[2] pointed out, they correspond to a set of variables with which a material balance can be written. They proved that by using the union of cutsets in a tree enumeration scheme, one can guarantee optimal solutions and, most importantly, reduce the computational time considerably. The virtue of this algorithm is that only meaningful measurements that contribute to the redundancy or observability of variables of interest (key variables) are added through the use of cutsets. The algorithm considerably cuts down computation time for large systems from months to days. To further reduce the computational time, Gala and Bagajewicz[27] proposed a decomposition algorithm. In this algorithm, the process graph is decomposed into subgraphs so as to reduce the number of cutsets in the candidate list, hence reducing the size of the tree. Cutsets spanning over subgraphs are generated at each node of the tree.

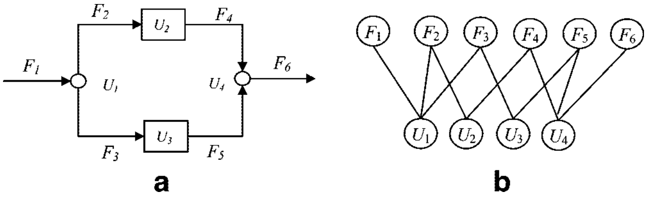

Figure 1. (a) Process digraph and (b) corresponding bipartite graph.

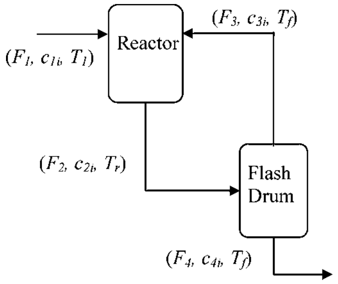

Figure 2. Example of nonlinear systems.

We now explore the possibilities and difficulties of using cutsets and tree enumeration with nonlinear problems.

3. Use of Cutsets in the Nonlinear Case

For linear systems where usually total flowrates are variables of interest, a stream (for which the flowrate is the variable) is connected to not more than two nodes: it is either an input to a unit, an output from a unit, or both an input to one unit and an output from another. As a result, a flowrate variable can always be represented by a stream in the process flowsheet, and the proper representation of a linear system is the process digraph.[29] Hence, for linear systems, the process constraint matrix that represents the overall material balance equations of the process is also the same as the incidence matrix that represents the connectivity of edges (streams) and vertices (nodes). This is not the case for nonlinear systems where a variable may occur in more than two equations, and therefore a digraph cannot be used to represent the system. A bipartite graph is, instead, the appropriate structural representation for nonlinear systems.[29] The digraph looks the same as the process flowsheet, the nodes are identified with equations, and the edges are the flows. In a bipartite graph, two rows of nodes are made, one for the variables and the other for the equations. This is illustrated for the linear system in Figure 1.

Consider a nonlinear process (Figure 2) consisting of an adiabatic reactor and an adiabatic flash drum. The nature of the

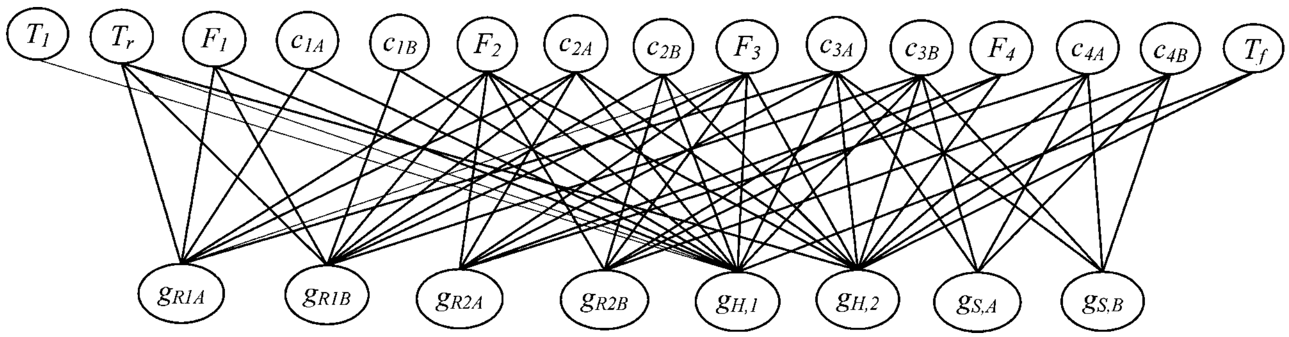

Figure 3. Bipartite graph of the nonlinear process example.

balance equations will determine the nonlinearity of the system. When only total flowrates (F_i) are variables of interest, the system is linear; when the variables concentration (C_i) and temperature (T_i) also need to be estimated, the system is nonlinear.

The corresponding equations are

$$g_{\mathrm{R1A}}(F_1, F_2, F_3, c_{1\mathrm{A}}, c_{2\mathrm{A}}, c_{3\mathrm{A}}, T_\mathrm{r}) = F_1c_{1\mathrm{A}} - F_2c_{2\mathrm{A}} + F_3c_{3\mathrm{A}} + \delta_\mathrm{A}(k_0\mathrm{e}^{-E/RT_\mathrm{r}}c_{2\mathrm{A}}^{\alpha})V = 0 \quad (2)$$

$$g_{\mathrm{R1B}}(F_1, F_2, F_3, c_{1\mathrm{B}}, c_{2\mathrm{B}}, c_{3\mathrm{B}}, c_{2\mathrm{A}}, T_\mathrm{r}) = F_1c_{1\mathrm{B}} - F_2c_{2\mathrm{B}} + F_3c_{3\mathrm{B}} + \delta_\mathrm{B}(k_0\mathrm{e}^{-E/RT_\mathrm{r}}c_{2\mathrm{A}}^{\alpha})V = 0 \quad (3)$$

$$g_{\mathrm{R2A}}(F_2, F_3, F_4, c_{2\mathrm{A}}, c_{3\mathrm{A}}, c_{4\mathrm{A}}) = F_2c_{2\mathrm{A}} - F_3c_{3\mathrm{A}} - F_4c_{4\mathrm{A}} = 0 \quad (4)$$

$$g_{\mathrm{R2B}}(F_2, F_3, F_4, c_{2\mathrm{B}}, c_{3\mathrm{B}}, c_{4\mathrm{B}}) = F_2c_{2\mathrm{B}} - F_3c_{3\mathrm{B}} - F_4c_{4\mathrm{B}} = 0 \quad (5)$$

$$g_{\mathrm{H1}}(F_1, F_2, F_3, c_{1\mathrm{A}}, c_{1\mathrm{B}}, c_{2\mathrm{A}}, c_{2\mathrm{B}}, c_{3\mathrm{A}}, c_{3\mathrm{B}}, T_1, T_r, T_\mathrm{f}) = F_1h_1(c_{1\mathrm{A}}, c_{1\mathrm{B}}, T_1) - F_2h_2(c_{2\mathrm{A}}, c_{2\mathrm{B}}, T_\mathrm{r}) + F_3h_3(c_{3\mathrm{A}}, c_{3\mathrm{B}}, T_\mathrm{f}) + (-\Delta H_{\mathrm{rxn}}(T_\mathrm{r}))k_0e^{-E/RT_\mathrm{r}}c_{2\mathrm{A}}^{\alpha}V = 0 \quad (6)$$

$$g_{\mathrm{H2}}(F_2, F_3, F_4, c_{2\mathrm{A}}, c_{2\mathrm{B}}, c_{3\mathrm{A}}, c_{3\mathrm{B}}, c_{4\mathrm{A}}, c_{4\mathrm{B}}, T_\mathrm{r}, T_\mathrm{f}) = F_2h_2(c_{2\mathrm{A}}, c_{2\mathrm{B}}, T_\mathrm{r}) - F_3h_3(c_{3\mathrm{A}}, c_{3\mathrm{B}}, T_\mathrm{f}) - F_4h_4(c_{4\mathrm{A}}, c_{4\mathrm{B}}, T_\mathrm{f}) - Q_{\mathrm{vap}} = 0 \quad (7)$$

$$g_{\mathrm{SA}}(F_1, c_{3\mathrm{A}}, c_{4\mathrm{A}}, c_{4\mathrm{B}}, T_\mathrm{f}) = c_{3\mathrm{A}}/\sum_i c_{3i} = K_i(c_{4\mathrm{A}}, c_{4\mathrm{B}}, T_\mathrm{f})c_{4\mathrm{A}}/\sum_i c_{4i} \quad (8)$$

$$g_{\mathrm{SB}}(F_1, c_{3\mathrm{A}}, c_{4\mathrm{A}}, c_{4\mathrm{B}}, T_\mathrm{f}) = c_{3\mathrm{B}}/\sum_i c_{3i} = K_i(c_{4\mathrm{A}}, c_{4\mathrm{B}}, T_\mathrm{f})c_{4\mathrm{A}}/\sum_i c_{4i} \quad (9)$$

Equations 2–5 are component balance equations (we assume that the system contains two components, A and B). Equations 6 and 7 are energy balance equations in which $h(...)$ are enthalpies. In these equations, we assume that the reactor is adiabatic and that a fixed known amount of heat (Q_{vap}) is removed in the flash. Finally, eqs 8 and 9 represent the vapor−liquid equilibrium relationship. The term $k_0\mathrm{e}^{-E/RT_\mathrm{r}}c_{2\mathrm{A}}^{\alpha}$ corresponds to reaction rate r_A (we assume only one irreversible reaction involving one reactant A). The corresponding bipartite graph is shown in Figure 3.

Cutsets of digraphs have been used by Gala and Bagajewicz[26,27] to design sensor networks for linear systems because (i) each cutset represents a material balance equation involving its elements (streams or variables), which is in turn directly connected to observability or redundancy of variables[2] and (ii) the properties of cutsets and procedures to enumerate all cutsets for linear systems are well-known. This procedure is illustrated in Figure 4 for a linear case and in Figure 5 for the corresponding ring sum operation in the context of bipartite graphs.

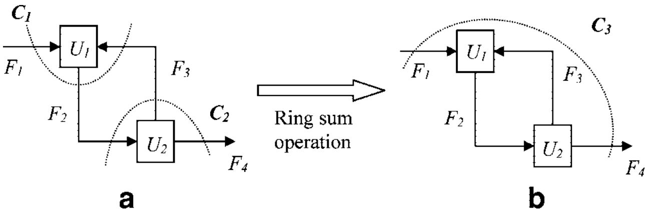

Figure 4. Process digraph and ring sum operation on cutsets of the linear system.

We will now show that the ring sum operation does not always apply properly to nonlinear cases. Consider equations g_{R1A} and g_{R2A}, which share two terms: $F_2c_{2\mathrm{A}}$ and $F_3c_{3\mathrm{A}}$ or four variables (F_2, $c_{2\mathrm{A}}$, F_3, and $c_{3\mathrm{A}}$). Substitution of $F_4c_{4\mathrm{A}} = F_2c_{2\mathrm{A}} - F_3c_{3\mathrm{A}}$ obtained from g_{R2A} into g_{R1A} renders the following equation

$$F_1c_{1\mathrm{A}} - F_4c_{4\mathrm{A}} + \delta_A(k_0\mathrm{e}^{-E/RT_\mathrm{r}}c_{2\mathrm{A}}^{\alpha})V = 0 \quad (10)$$

The ring sum operation results in these four variables being eliminated as seen in Figure 6.

We note first that the ring sum of these two cutsets leads to the elimination of all four variables that are in the intersection, including $c_{2\mathrm{A}}$. But, unless the reaction is zero order, $c_{2\mathrm{A}}$ is still present in the merged eq 10. Thus, equation merging or variable substitution generates in this case a result that is different from the ring sum operation. We assume here that variable substitution can take place when one variable can be explicitly expressed as a function of others in one equation and formally substituted in a second equation. Equation merging is a term we formally reserve for the case in which the substitution cannot be performed analytically, so the assumption is made that by considering those two equations simultaneously with all the other variables known, one can (numerically or otherwise) solve for the variable in question.

Thus, for nonlinear systems one needs to depart from using the ring sum of cutsets strictly and look for an equivalent procedure. Such a procedure would be the equivalent of finding an alternative operation to the ring sum, perhaps adding a criterion as to what variables are eliminated after the union is made. Indeed, as stated above, cutsets are equations in the linear case, and since the ring sum is equivalent to taking two equations and generating a third, thus eliminating one (or more) variables, we resort to the same concept, but we use linearized equations.

4. Automatic Generation of All Equations

Knowing now what needs to be mimicked (the variable substitution process), we now prove that the determination of

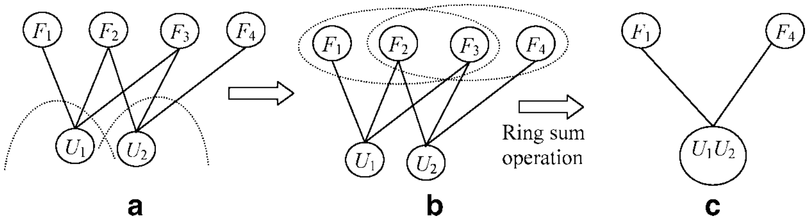

Figure 5. Bipartite graph and ring sum operation on cutsets of the linear system.

what variables of the intersection have to be eliminated can be done by inspection but also automatically using Gaussian elimination on the linearized equations. We do this by proving the following claims first.

Claim 1 (Necessary Condition). Consider two nonlinear equations

$$f_1(x_1, x_2, ..., x_k, x_{k+1}, ..., x_m) = 0 \quad (11)$$

$$f_2(x_k, x_{k+1}, ..., x_m, x_{m+1}, ..., x_n) = 0 \quad (12)$$

If equation merging or variable substitution is performed targeting variable x_k, and a set of other variables, namely, x_{k+1},..., x_m, which are considered consecutive without loss of generality, are also eliminated, then the following is true:

(a) The partial derivatives with respect to these eliminated variables in both equations are equal, that is,

$$\frac{\partial f_1}{\partial x_t} = \frac{\partial f_2}{\partial x_t} \qquad \forall\, t = k+1, ..., m \quad (13)$$

(b) Variable substitution in the linearized system also eliminates the same variables.

Proof. Assume first (without loss of generality) that f_1 and f_2 can be expressed in terms of x_k as follows:

$$f_1(x_1, x_2, ..., x_k, x_{k+1}, ..., x_m) = \Sigma_i r_{1i}(x_1, x_2, ..., x_{k-1}, x_{k+1}, ..., x_m) + x_k = 0 \quad (14)$$

$$f_2(x_1, x_2, ..., x_{k-1}, x_{k+1}, ..., x_m) = \Sigma_i r_{2i}(x_1, x_2, ..., x_{k-1}, x_{k+1}, ..., x_m) + x_k = 0 \quad (15)$$

In other words, x_k can be isolated. Then, substitution of x_k in equation f_2 renders

$$f_{12}(x_1, x_2, ..., x_{k-1}, x_{k+1}, ..., x_m, x_{m+1}, ..., x_n) = \Sigma_i r_{1i}(x_1, x_2, ..., x_{k-1}, x_{k+1}, ..., x_m) - \Sigma_i r_{2i}(x_1, x_2, ..., x_{k-1}, x_{k+1}, ..., x_m) = 0 \quad (16)$$

For variables to be eliminated, pairs of r_{1i} and r_{2i} need to be exactly the same expressions with the same combinations of variables so that they cancel. Without loss of generality, assume now these variables are in both equations in the terms r_{11} and r_{21} only, that is,

$$r_{11}(x_1, x_2, ..., x_{k-1}, x_{k+1}, ..., x_m) = r_{21}(x_1, x_2, ..., x_{k-1}, x_{k+1}, ..., x_m) = 0 \quad (17)$$

Because (and only because) these terms disappear from the final version of 16, we can say that they *are* the same expression, which, in turn, allows us to say the derivatives are also formally the same expression. This proves part a.

Proving part b is now easy. Indeed, by linearizing f_1 and f_2, one obtains

$$\sum_{j \neq k} a_{1j} x_j + x_k = 0 \quad (18)$$

$$\sum_{j \neq k} a_{2j} x_j + x_k = 0 \quad (19)$$

where $a_{ki} = \{\partial[\Sigma_i r_k]\}/\partial x_i$. Substituting x_k obtained from eq 18 into eq 19, one obtains

$$\sum_{j \neq k} (a_{2j} - a_{1j}) x_j = 0 \quad (20)$$

Thus, if any variable is to be eliminated, say, x_{k+1}, then $a_{2k+1} = a_{1k+1}$, which is the same as eq 13. *Q.E.D.*

Claim 2 (Sufficient Condition). Consider the linearized eqs 18 and 19. If Gaussian elimination is performed between these two equations, aside from variable x_k the set of other variables, namely, x_{k+1},..., x_m, is also eliminated, then the same variables will be eliminated if equation merging or variable substitution is performed on eqs 14 and 13, provided that eq 13 holds symbolically (i.e., they are the same symbolic expression), not only numerically.

Proof. The proof is straightforward: If a variable is eliminated, say x_{k+1}, then $a_{2k+1} = a_{1k+1}$, which is the same as 13 numerically. We only need to make sure that eq 17 holds, and this is true only if eq 13 or 17 holds true symbolically. *Q.E.D.*

Corollary. Both claims are valid if explicit expressions in terms of x_k cannot be obtained.

Proof. In this case, we would have

$$f_1(x_1, x_2, ..., x_k, x_{k+1}, ..., x_m) = 0 \rightarrow \Sigma_i r_{1i}(x_1, x_2, ..., x_{k-1}, x_{k+1}, ..., x_m) + z(x_1, x_2, ...x_k, x_{k-1}, x_{k+1}, ..., x_m) = 0 \quad (21)$$

Thus, if x_k is to be formally eliminated from both equations, then the term $z(x_1, x_2,..., x_{k,}, x_{k-1}, x_{k+1},... x_m)$ needs to exactly appear in both equations. With this the proof can be continued exactly as before. *Q.E.D.*

We can now present the procedure to find and enumerate all equations of the problem:

(1) Linearize the process model to arrive at the linearized model written in the matrix form $\mathbf{A}x = b$, where $\mathbf{A}$ is the process constraint matrix.

(2) For all pairs of equations, perform the Gaussian elimination operation to find new equations and put them at the end of the list of equations. If any pair of equations has more than one common variable, all possible new equations are to be found by choosing different variables to eliminate. Note that only combinations of equations with at least one common variable are performed.

(3) If a resulting new equation is the same as any equation already in the list, disregard that equation.

Because all combinations of original equations are considered, this procedure guarantees that all possible new equations are found. Combinations between a new equation and an original equation or between a new equation and another new equation are not necessary because they can be obtained by combining original equations.

There are two steps in the design procedure that call for the linearization of the nonlinear equations around nominal operating conditions:

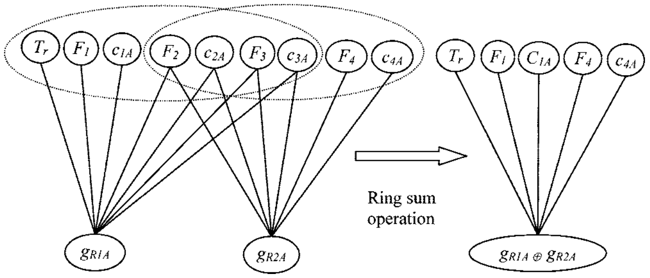

Figure 6. Ring sum operation on cutsets of the bipartite graph.

Table 1. Summary of Proposed Tree Search Methods

method	search strategy	base unit	decomposed
All Variables	forward	measurements	no
All Equations	forward	equations	no
Decomposed Equations	forward	equations	yes
Inverted All Variables	reverse	measurements	no

(i) Finding new equations from pairs of "original" equations using the variable substitution or equivalently the Gaussian elimination operation.

(ii) Solving the data reconciliation problem using the data provided by the sensor network chosen, in which an analytical solution is obtainable only when the model is linear or linearized, in order to check whether the candidate sensor network satisfies the design specifications.

Changing operating conditions would not cause any effect in the equation generating step (the resulting equations are unchanged, only the coefficients in the equations change), but it may have an effect in the step of checking design specifications: for example, a feasible solution can become infeasible if the operating window moves to another region. As a result, different regions of operating conditions may lead to different optimal solutions, and the obtained optimal solution is guaranteed to be valid only within the current operating windows. Designing an optimal sensor network that is valid for a wide range of operating conditions requires a new problem formulation and a tailored computational method, which is beyond the scope of this article. In the examples section, however, we provide a brief discussion of the sensitivity of the solution as the process variables fluctuate around their nominal values.

5. Equation-Based Tree Search Algorithm

The tree enumeration algorithm using equations for the design of nonlinear sensor networks is the same as the one used by Gala and Bagajewicz,[26] except that instead of using cutsets, we use equations. We just include it here for completeness.

1. Find all the equations of the problem using the procedure described above.

2 Pick up only the equations containing key variables (called candidate equations) because other equations (not containing key variables) do not contribute to the observability or redundancy of key variables.

3. Sort these candidate equations in ascending order of their cost (the cost of an equation is equal to the sum of the costs of the sensors used to measure variables contained in that equation).

4. Start with the root node with no equation being added, that is, $e = \{0, 0, 0,...\}$, trivially infeasible.

5. Using the branch first rule, develop each branch by making one element of "e" active and adding one candidate equation at a time which is chosen from the remaining equations using a branching criterion.

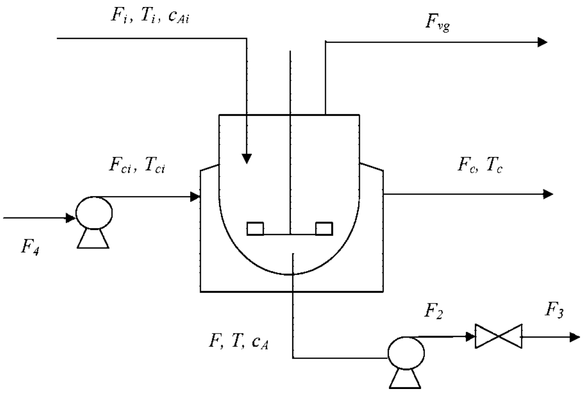

Figure 7. CSTR problem.

Table 2. Nominal Operating Condition for the CSTR Example

variable	F_i	F_c	F_{vg}	F	F_2	F_3	F_4
value	40	56.626	10.614	40	40	40	56.626
variable	c_{Ai}	c_A	T	T_i	T_c	T_{ci}	
value	0.5	0.2345	600	530	590.51	530	

6. While performing the branching criteria, if any set of equations has already been evaluated in previous nodes, that node is not continued. This occurs frequently because one set of measurements can be a result of the union of different combinations of equations.

7. This is continued until the stopping criterion is met. In such case, the algorithm backs up two levels and develops the next branch.

Branching Criterion. While exploring the tree from one node to the other, either going down the tree or exploring the sister node, the newly added equation is chosen in such a way that the cost obtained by its union with the existing active equations is minimal.

Stopping Criterion. Because adding an equation always increases the cost, whenever a feasible node is found (one that satisfies all the constraints of the problem), the tree is not explored further down nor along any sister branch.

To overcome the computational limitations of the above procedure, Gala and Bagajewicz[27] proposed the "decomposition of process graph network" algorithm to reduce computation time. The algorithm still makes use of cutsets to find the optimum sensor network, but the process graph is decomposed into subgraphs so as to reduce the number of original cutsets, hence, reducing the size of the tree. While exploring the tree, if cutsets from different subgraphs are used in a node of the tree, the ring sum operation on those cutsets is performed to generate all the cutsets that are missing, and then the union operation among the resulting cutsets plus the originals is performed. The principles behind the technique, the benefits, and the calculation procedure of this decomposition method for the design of linear sensor network were explained by Gala and Bagajewicz.[27] For nonlinear systems, we use the same technique as Gala and Bagajewicz[27] with some modifications in the calculation procedure. These modifications are (i) the ring sum operation on cutsets is replaced by our variable elimination operation on equations and (ii) the decomposition is performed on the bipartite graph of the nonlinear system instead of the process digraph of linear system. We also use the same branching and stopping criterion.

6. Inverted Tree Strategy

If a large number of key variables (those whose values are of interest) and/or good level of precision and residual precision

Chart 1

$$A_{nl} = \begin{array}{ccccccccccccc|c} F_i & c_{Ai} & c_A & T & T_i & T_c & F_c & T_{ci} & F_{vg} & F & F_2 & F_3 & F_4 & \\ \hline -0.00531 & -0.8333 & 1.7763 & 0.00923 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_1 \\ 1.4583 & 0 & -754.4 & 5.9503 & -0.8333 & -125 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_2 \\ 0 & 0 & 0 & -93.8067 & 0 & 108.5 & 15.7169 & -14.708 & 0 & 0 & 0 & 0 & 0 & e_3 \\ 0 & 0 & -45.2612 & -0.443 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & e_4 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & e_5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & e_6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & e_7 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & e_8 \end{array}$$

are required, a large number of sensors needs to be used to meet the requirements. This is accentuated if error detectability and resilience requirements are added. In such design cases, the "forward" tree search methods (the equations-based method presented above as well as the tree search methods presented in Bagajewicz[13] and Gala and Bagajewicz[27]) exhibit the following problems:

(i) The number of active elements in feasible nodes is large; that is, the search procedure needs to explore deeper down into the tree before it finds a feasible solution.

(ii) Consequently, the number of nodes explored is large and the computational time is long. Moreover, for the equation-based tree search methods, a large number of key variables leads to a large number of equations that contain at least one key variable (i.e., large tree size).

(iii)The number of feasible nodes is low, and they are all located deep in the tree toward the end of it.

To ameliorate this shortcoming (having to explore the tree very deep), we propose an inverted tree search method. The idea behind this method is to explore the tree in the *reverse* direction, that is, to start with a root node containing *all* sensors and continue *removing* sensors when going up the tree until an infeasible node is found (stopping criterion). Because the level of the feasible nodes explored by the tree search is low, the number of nodes explored is reduced, which results in a shorter computation time. The same thing is also argued if equations are used instead of sensors (measurements).

There are two "forward" tree search methods as discussed above: one uses list of equations/cutsets, and the other uses a list of measurements. We investigate the reverse versions of these two tree search methods. The inverted tree search using the list of measurements is described next:

1. Start tree search with the root node containing all sensors, an automatically feasible node.

2. Remove sensors out of root node by using the tree enumeration algorithm and a branching criterion, which is to remove the most expensive sensor among all sensors in the current node so that the sensors cost is minimized.

3. Always start developing branches with feasible nodes containing a large number of sensors. The number of sensors in nodes decreases, and cost is reduced when going up the tree.

4. Stop going up (stop removing sensors) when the current node becomes infeasible (stopping criterion). If continuing to go up, the cost is reduced but the node is infeasible.

An inverted tree search using equations (root node containing *all* equations and continues *removing* equations when going up the tree) is much less efficient, in principle, because of two problems that lead to longer computation time:

(i) The number of equations is large (much larger than the number of variables); hence, the tree depth is larger.

(ii) The search needs to explore further up into the tree before it finds an infeasible node and stop. The reason for this is that the union of only a small number of equations can result in the sensor network containing *all* sensors. Thus, if only a small number of equations is removed from the root node, the resulting sensor network (obtained as a union of equations in the current node) usually still contains *all* sensors. As a result, a significant number of equations needs to be removed before the number of sensors in the resulting sensor network is reduced and the node becomes infeasible.

In summary, in this paper, three methods are proposed to solve the nonlinear sensor network design problem: the inverted tree search using the list of measurements (referred to by the short name of "Inverted All Variables" method) and the "forward" equations-based tree search methods that has two versions: (i) without decomposition (referred to as "All Equations" method) and (ii) with decomposition (referred to as "Decomposed Equations" method). The characteristics of these three methods together with the forward tree search method using list of measurements[13] (referred to as "All Variables" method), which is used to validate the optimality of obtained solutions, are summarized in Table 1.

All four of these methods guarantees optimality of solution because, as a branch and bound method, they search the whole space of variables where all possible sensor configurations are investigated. Indeed, the stopping criterion helps eliminate the regions not containing the optimal solution, and the branching criterion helps locate the optimal solution faster. In fact, all the design case studies presented below show that all four methods find the same solution, which is guaranteed to be the optimal solution. Only in the case that the computation process has to be terminated because it has run for too long is the obtained solution not guaranteed to be the optimal solution. However, thanks to its equations-based approach, even in the case that the computation process is terminated halfway such that the current best solution (the incumbent) found by the equations-based methods is not the true optimum, it is very near the optimal solution as demonstrated in example 3 shown below.

7. Examples

The proposed equations-based methods and the "Inverted All Variables" method guarantee optimality; their computational efficiency is tested using the following examples. The "All Variables" method[13] is used to validate the optimality of the solutions obtained by the proposed methods. The proposed algorithms were implemented in a Fortran program running on a 2.8 GHz Intel Pentium, 1028 MB RAM PC computer.

Three examples are considered: a CSTR process (small scale problem), a mineral flotation process (middle scale problem), and the TE process (large scale problem). Based on our experience, we qualitatively classify three types of problems that can be solved by our sensor network design program using the number of variables involved: (i) small scale, 1−18

Table 3. New Equations for the CSTR Example Obtained from the Combination of e_1 and e_2

eq	
	(a) Variables Involved
e_9	$\{c_{Ai}, c_A, T, T_i, T_c\}$
e_{10}	$\{F_i, c_{Ai}, T, T_i, T_c\}$
e_{11}	$\{F_i, c_{Ai}, c_A, T_i, T_c\}$
	(b) Expressions
e_9	$c_dc_Ak_0e^{-E/RT}(T_i - T)/(c_{Ai} - c_A) + c_dc_Ak_0e^{-E/RT}(-\Delta H)/\rho C_P - UA(T-T_c)/V\rho C_P = 0$
e_{10}	$F_i/V(T_i - T) + c_dk_0e^{-E/RT}(-\Delta H)F_ic_{Ai}/\rho C_P(F_i+Vc_dk_0e^{-E/RT}) - UA(T-T_c)/V\rho C_P = 0$
e_{11}	$F_i/V(T_i - T) + F_i/V(c_{Ai} - c_A)(-\Delta H)/\rho C_P - UA/V\rho C_P \times \left(-E/R \ln\left(F_i/V(c_{Ai}-c_A)/c_dc_Ak_0\right)\right) = 0$
	(c) Linearized Expressions
e_9	$-0.833c_{Ai} - 0.972c_A + 0.031T - 0.003T_i - 0.456T_c$
e_{10}	$-0.002F_i - 0.83c_{Ai} + 0.0232T - 0.002T_i - 0.294T_c$
e_{11}	$-0.0076F_i - 0.833c_{Ai} + 2.946c_A + 0.0013T_i + 0.194T_c$

variables; (ii) middle scale, 19−39 variables; and (iii) large scale, 40 variables and above. As usual, these are heuristic observations, and although the number of variables is indicative of size, as we shall see below, the tightness of the specifications may make the same size problem be solved much faster/slower.

7.1. Example 1: Simple CSTR. Consider the CSTR process which was introduced by Bhushan and Rengaswamy[6] and is given in Figure 7.

The variables of interest are F_i, c_{Ai}, c_A, T, T_i, T_c, F_c, T_{ci}, F_{vg}, F, F_2, F_3, and F_4. There are five equations (including both mass and energy balances) written around the reactor and its jacket (eqs 22–26) and three mass balance equations written for pumps and valves (eqs 27–29).

$$e_1(F_i, c_{Ai}, c_A, T) = \frac{F_i}{V}(c_{Ai} - c_A) - c_dc_Ak_0e^{-E/RT} = 0 \quad (22)$$

$$e_2(F_i, c_A, T, T_i, T_c) = \frac{F_i}{V}(T_i - T) + \frac{c_dc_Ak_0e^{-E/RT}(-\Delta H)}{\rho C_p} - \frac{UA(T - T_c)}{V\rho C_p} = 0 \quad (23)$$

$$e_3(T, T_c, F_c, T_{ci}) = \frac{F_c}{V_j}(T_{ci} - T_c) + \frac{UA(T - T_c)}{V_j\rho_jC_{pj}} = 0 \quad (24)$$

$$e_4(c_A, T, F_{vg}) = c_dc_Ak_0e^{-E/RT}V - F_{vg} = 0 \quad (25)$$

$$e_5(F_i, F) = F_i - F = 0 \quad (26)$$

$$e_6(F_2, F_3) = F_3 - F_2 = 0 \quad (27)$$

$$e_7(F, F_2) = F_2 - F = 0 \quad (28)$$

$$e_8(F_c, F_4) = F_4 - F_c = 0 \quad (29)$$

The nominal operation conditions are given in Table 2 (value of flowrate is given in ft^3/h; temperature, °R; concentration, lb·mol/ft^3).

The linearized model matrix is shown in Chart 1.

We now illustrate the process of generation of new equations. Take for example the original equations e_1 and e_2, which have three common variables F_i, c_A, and T. Three new equations result from the three possible Gaussian elimination operations: $e_9 = \{c_{Ai}, c_A, T, T_i, T_c\}$ obtained by eliminating the common variable F_i, $e_{10} = \{F_i, c_{Ai}, T, T_i, T_c\}$ obtained by eliminating the common variable c_A, and $e_{11} = \{F_i, c_{Ai}, c_A, T_i, T_c\}$ obtained by eliminating the common variable T. These three new equations are shown in Table 3.

It is clear from this example that the number of equations in nonlinear systems is much larger than the number of cutsets in linear systems of the same flowsheet size, not only because more equations are written for each unit but also because any combination of cutsets results in just one new cutset in linear systems, while several new equations can be obtained from a combination of equations in nonlinear systems as illustrated above.

Next, we show case studies for this process using the proposed algorithms. For the Decomposed Equations method, a single decomposition is performed; that is, the graph is decomposed into two subgraphs corresponding to two subset constraint matrices, one that contains rows 1−4 and one that contains rows 5−8 (the cutting is done between rows 4 and 5). This is shown in Figure 8 where the original bipartite graph is decomposed into two subgraphs (A and B), each containing 4 nodes (4 original equations). The total number of equations obtained from all eight original equations is 88, while the number of equations obtained from the first four original equations (i.e., subgraph A) is 28 and from the last four original equations (i.e., subgraph B) is 6. Thus, in the decomposition method, the total number of equations (i.e., the tree size) is 34 (= 28 plus 6) instead of 88. The rest of the equations are found while exploring the tree by using variable elimination operation on any pair of equations originated from any two different subgraphs.

The costs of sensors that measure variables V_1, V_2,..., V_{13} are 100, 270, 300, 50, 55, 60, 105, 45, 85, 90, 95, 80, and 82, respectively. The sensor precisions are 1% (for all sensors).

Three design cases are considered corresponding to three levels of design specification: low (CSTR1), moderate (CSTR2), and high specification (CSTR3), which are shown in Table 4. In Table 4 as well as similar tables in the other two examples, rows 2 to 6 show detail of specifications for each design case; the stated threshold values (rows 5, 6) are applied to all the key variables listed in row 3, and any key variable with specific threshold will be mentioned separately. Rows 7 and 8 show the optimal solution obtained by the four methods, which include two types of information: the variables to measure (row 7) and the total sensor cost (row 8).

The computation time and the number of nodes explored in the four methods are shown in Table 5.

When comparing the computation times, the equation-based method is sometimes faster than that of the All Variables method because of the smaller number of nodes explored and sometimes slower because the equation-based method initially requires finding all the equations of the system and requires checking the branching criterion in every node of the tree. The computation time of the Decomposed Equations method is shorter than that of the All Equations method as expected.

The method with shortest computational time in this example is the "Inverted All Variables" method, which can be seen to be faster than its "forward" counterpart, the "All Variables" method, especially for the high specification design case (CSTR3).

Because in this example the equation-based methods do not offer much advantage in terms of reduced number of nodes explored and requires extra time for performing branching at any node, the real test for the advantage of the equation-based methods has to come from applying them to a larger example.

We briefly discuss the sensitivity of the obtained solution before we present the next example. As we linearize the nonlinear equations around the steady-state values of the process variables, the obtained solution is valid only within the current operating window. Take for example the design

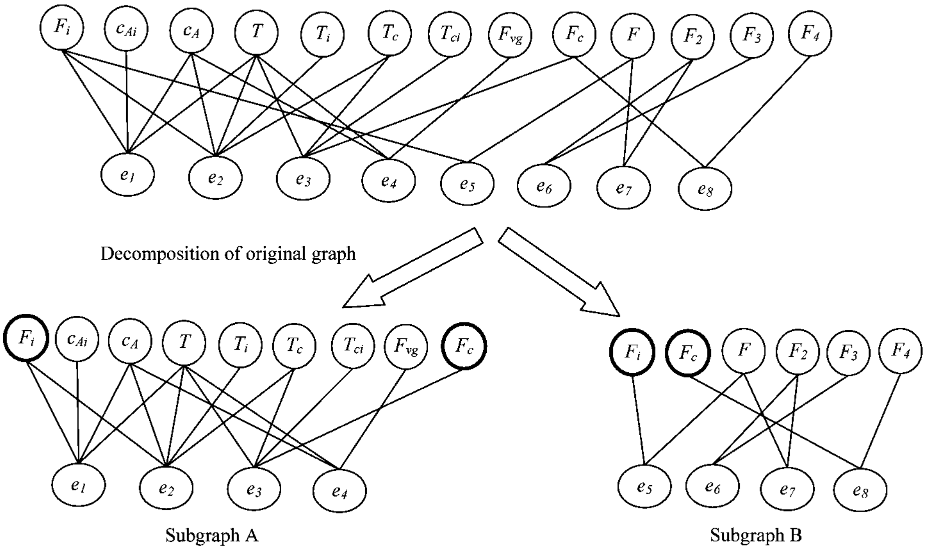

Figure 8. Bipartite graph of the CSTR example with decomposition (common variables are shown in thicker circles).

Table 4. Design Case Studies for the CSTR Example

case study	CSTR1 (low spec.)	CSTR2 (moderate spec.)	CSTR3 (high spec.)
no. of key variables	3	4	8
key variables	c_A, T, F	c_A, T, T_{ci}, F	c_{Ai}, c_A, T, T_c, F_c, T_{ci}, F, F_3
requirement	observability	redundancy	observability
precision thresholds	0.95%	1.5%	1.5%
residual precision thresholds		2.5%	2.5%
measured variables	c_{Ai}, c_A, F_{vg}, F_3	c_{Ai}, c_A, T, T_i, T_{ci}, F, F_3, F_4	c_{Ai}, c_A, T, T_i, T_c, F_c, T_{ci}, F, F_3, F_4
sensor cost	735	972	1137

Table 5. Results for the CSTR Example

case study		CSTR1 (low spec.)	CSTR2 (moderate spec.)	CSTR3 (high spec.)
All-Equations (no decomposition)	total computation time	8 s	26 s	27 s
	computation time to generate equations	1 s	1 s	1 s
	number of equations generated	85	87	87
	number of nodes explored	1611	4695	7640
Decomposed Equations (two subgraphs)	total computation time	4 s	8 s	14 s
	computation time to generate equations	<1 s	<1 s	<1 s
	number of equations generated	33	33	34
	number of nodes explored	1737	2914	5456
All Variables	computation time	3 s	5 s	6 s
	number of nodes explored	4653	7088	8102
Inverted All Variables	computation time	2 s	2 s	1 s
	number of nodes explored	2520	682	117

Table 6. Nominal Operation Condition for Mineral Flotation Process

stream	1	2	3	4	5	6	7	8
F_i (kmol/h)	100	92.67	91.57	84.48	7.33	8.43	7.09	1.1
C_{iA} (% mol)	0.019	0.0045	0.0013	0.001	0.2027	0.2116	0.0051	0.2713
C_{iB} (% mol)	0.0456	0.0437	0.0442	0.0041	0.069	0.0495	0.5227	0.001

Table 7. Sensor Costs for Mineral Flotation Process Example

streams	1	2	3	4	5	6	7	8
F_i	50	55	45	60	40	48	52	58
C_{iA}	300	310	240	260	250	360	320	335
C_{iB}	290	350	330	340	280	270	295	275

case CSTR1: if the steady-state values of the process variables (except the reaction temperature) change within 40% of the nominal values, the precision values of the estimators change up to 95% of the precision values in the base case (corresponding to the nominal values) but still satisfy the design specifications; that is, the obtained solution is valid when the fluctuation is less than 40% of the nominal value. If the fluctuation is more than 40% of the nominal values (except the reaction temperature), the new optimal solution is to measure 7 variables with cost of 737. The obtained solution is more sensitive to the variation of the reaction temperature, an important variable whose fluctuation significantly affects all other variables in the process. The variation range of the reaction temperature within which the obtained solution is valid is 26% of the nominal value. These results point out that the solution is valid within the normal variation range of process variables (30% of the nominal values), but it is not valid for a wider range of process operating conditions.

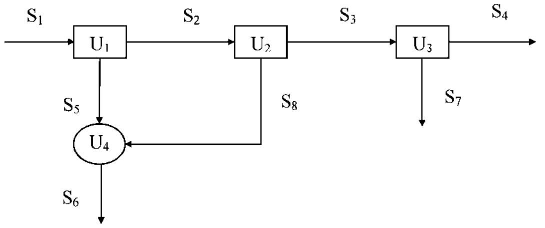

Figure 9. Mineral flotation process.

7.2. Example 2. Mineral flotation Process.

Consider a middle scale process, the mineral flotation process introduced by Smith and Ichiyen[30] shown in Figure 9.

The process consists of three flotation cells (separators) and a mixer. Each stream consists of two minerals, copper (component A) and zinc (component B), in addition to gangue material. The total flowrate F and the composition of copper C_A and zinc C_B of all streams are variables of interest, so the total number of variables under consideration is 24 (8 flowrates and 16 compositions). Let us assume that each variable can be measured separately by a sensor. The process model consists of three types of material balance equations: the total flowrate balance, the copper component (A) flowrate balance, and the zinc component (B) component balance. These three types of balance equations are written for unit 1 next. Balance equations for other units can be written in the same fashion:

$$F_1 - F_2 - F_5 = 0 \tag{30}$$

$$F_1C_{1A} - F_2C_{2A} - F_5C_{5A} = 0 \tag{31}$$

$$F_1C_{1B} - F_2C_{2B} - F_5C_{5B} = 0 \tag{32}$$

The total number of original balance equations is 12 (3 per unit). The component balance equations are nonlinear; hence, the system is nonlinear (it is a bilinear system). The nominal operating condition is given in Table 6 (taken from Narasimhan and Jordache[31]).

When the balance equations are linearized, the process model can be written in the following form $\mathbf{A}x = b$ (see Chart 2 for the definition of $\mathbf{A}$). The first four rows in the constraint matrix A corresponds to the total flow balances, row 5 to row 8 represent copper component balances, and the rest corresponds to zinc component balances. All sensor precisions are 2%. The sensor costs are given in Table 7.

Three design cases at three different levels of design specification are considered: low (MFP1), moderate (MFP2), and high specification (MFP3). They are shown in Table 8.

In all case studies, the decomposition is made by operating directly on the constraint matrix; that is, the constraint matrix is "cut" into three subset matrices: {row 1 to row 4}, {row 5 to row 8}, and {row 9 to row 12}. The submatrices for these systems are indicated by dotted rectangles in matrix **A** (Chart 3). The common variables between the subset matrices (called connecting streams in the case of linear systems) are the eight flowrate variables: from F_1 to F_8. The computation time and number of nodes explored are shown in Table 9.

In the case study MFP1, the number of equations containing at least one of the four key variables {F_1, F_7, C_{1A}, C_{7B}} in the equations-based tree search without decomposition (All Equations method) is 2225. When the decomposition technique is used, the number of equations containing at least one key variable reduces significantly to 27, which is the main reason why the computation time reduces remarkably to less than 1 min in both design cases. For comparison, in the linear Madron example solved by Gala and Bagajewicz,[26] which has the same scale as our mineral example problem with 5 out of 24 variables required to be observable, the number of cutsets (depth of the tree) is 154. Although the All Variables method explores a lot more nodes than the All Equations method, the time is considerably smaller because the branching criterion in the All Equations method requires expensive computation to pick up the equation leading to the minimum cost sensor network among roughly 2210−2220 candidates (= total number of equations minus the number of equations active in the current node).

Chart 2

$$A = \begin{array}{cccccccccccccccccccccccc}
F_1 & F_2 & F_3 & F_4 & F_5 & F_6 & F_7 & F_8 & C_{1A} & C_{1B} & C_{2A} & C_{2B} & C_{3A} & C_{3B} & C_{4A} & C_{4B} & C_{5A} & C_{5B} & C_{6A} & C_{6B} & C_{7A} & C_{7B} & C_{8A} & C_{8B} \\
1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.02 & -0.005 & 0 & 0 & -0.203 & 0 & 0 & 0 & 100 & 0 & -92.67 & 0 & 0 & 0 & 0 & 0 & -7.33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.005 & -0.001 & 0 & 0 & 0 & 0 & -0.271 & 0 & 0 & 92.67 & 0 & -91.57 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.1 & 0 \\
0 & 0 & 0.001 & -0.001 & 0 & 0 & -0.005 & 0 & 0 & 0 & 0 & 0 & 91.57 & 0 & -84.48 & 0 & 0 & 0 & 0 & 0 & -7.09 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.203 & -0.212 & 0 & 0.271 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7.33 & 0 & -8.43 & 0 & 0 & 0 & 1.1 & 0 \\
0.05 & -0.044 & 0 & 0 & -0.069 & 0 & 0 & 0 & 0 & 100 & 0 & -92.67 & 0 & 0 & 0 & 0 & 0 & -7.33 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.044 & -0.044 & 0 & 0 & 0 & 0 & -0.001 & 0 & 0 & 0 & 92.67 & 0 & -91.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.1 \\
0 & 0 & 0.044 & -0.004 & 0 & 0 & -0.523 & 0 & 0 & 0 & 0 & 0 & 0 & 91.6 & 0 & -84.5 & 0 & 0 & 0 & 0 & 0 & -7.09 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.069 & -0.05 & 0 & 0.001 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7.33 & 0 & -8.43 & 0 & 0 & 0 & 1.1
\end{array}$$

Table 8. Design Case Studies for the Mineral Flotation Process Example

case study	MFP1 (low spec.)	MFP2 (moderate spec.)	MFP3 (high spec.)
no. of key variables	4	4	12
key variables	F_1, C_{1A}, F_7, C_{7B}	F_1,C_{1A}, F_7, C_{7B}	F_1, F_4, F_6, C_{1A}, C_{1B}, F_7, C_{4A}, C_{4B}, C_{6A}, C_{6B}, C_{7A}, C_{7B}
requirement	observability	redundancy	observability
precision thresholds	1.5% (F_1, C_{1A}) 2% (F_7, C_{7B})	1.5% (F_1, C_{1A}) 2% (F_7, C_{7B})	1.5% (F_1, F_4, F_6, C_{1A}, C_{1B}) 2% (F_7, C_{4A}, C_{4B}, C_{6A}, C_{6B}, C_{7A}, C_{7B})
residual precision thresholds		5% for F_1, F_7 only	
measured variables	F_1, F_3, F_5, F_6, F_7, F_8, C_{1A}, C_{2A}, C_{5A}, C_{7B}	F_1, F_3, F_5, F_6, F_7, F_8, C_{1A}, C_{2A}, C_{3B}, C_{4B}, C_{5A}, C_{7B}	F_1, F_3, F_5, F_6, F_7, F_8, C_{1A}, C_{2A}, C_{3B}, C_{4A}, C_{4B}, C_{5A}, C_{6B}, C_{7A}, C_{7B}
sensor cost	1448	2118	2968

Chart 3

$$A = \begin{array}{cccccccccccccccccccccccc}
F_1 & F_2 & F_3 & F_4 & F_5 & F_6 & F_7 & F_8 & C_{1A} & C_{1B} & C_{2A} & C_{2B} & C_{3A} & C_{3B} & C_{4A} & C_{4B} & C_{5A} & C_{5B} & C_{6A} & C_{6B} & C_{7A} & C_{7B} & C_{8A} & C_{8B} \\
1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.02 & -0.005 & 0 & 0 & -0.203 & 0 & 0 & 0 & 100 & 0 & -92.67 & 0 & 0 & 0 & 0 & 0 & -7.33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.005 & -0.001 & 0 & 0 & 0 & 0 & -0.271 & 0 & 0 & 92.67 & 0 & -91.57 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.1 & 0 \\
0 & 0 & 0.001 & -0.001 & 0 & 0 & -0.005 & 0 & 0 & 0 & 0 & 0 & 91.57 & 0 & -84.48 & 0 & 0 & 0 & 0 & 0 & -7.09 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.203 & -0.212 & 0 & 0.271 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7.33 & 0 & -8.43 & 0 & 0 & 0 & 1.1 & 0 \\
0.05 & -0.044 & 0 & 0 & -0.069 & 0 & 0 & 0 & 0 & 100 & 0 & -92.67 & 0 & 0 & 0 & 0 & 0 & -7.33 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.044 & -0.044 & 0 & 0 & 0 & 0 & -0.001 & 0 & 0 & 0 & 92.67 & 0 & -91.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.1 \\
0 & 0 & 0.044 & -0.004 & 0 & 0 & -0.523 & 0 & 0 & 0 & 0 & 0 & 0 & 91.6 & 0 & -84.5 & 0 & 0 & 0 & 0 & 0 & -7.09 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.069 & -0.05 & 0 & 0.001 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7.33 & 0 & -8.43 & 0 & 0 & 0 & 1.1
\end{array}$$

Table 9. Results for Mineral Flotation Process Example

case study		MFP1 (low spec.)	MFP2 (moderate spec.)	MFP3 (high spec.)
All-Equations	computation time	6 h, 22 min	45 h, 44 min	not used
	computation time to generate equations	2 s	2 s	not used
	number of equations generated	2225	2225	not used
	number of nodes explored	5645	35 289	not used
decomposed eqs (3 subgraphs)	computation time	13 s	40 s	1 h, 29 min
	computation time to generate equations	<1 s	<1 s	<1 s
	number of equations generated	27	27	33
	number of nodes explored	5077	13 622	200 245
All Variables	computation time	23 min, 41 s	2 h, 16 min	7 h, 35 min
	number of nodes explored	529 130	3 743 327	12 366 120
Inverted All Variables	computation time	17 min, 20 s	11 min, 18 s	49 s
	number of nodes explored	376 432	130 733	19 722

The larger tree size in this nonlinear example leads to much longer computation time because of two main reasons:

(i) A larger tree depth requires an exponentially longer computation time to explore the tree.

(ii) A larger number of equations requires a longer time to perform the branching criterion in a node (picking the equation leading to the minimum-cost sensor network among all candidates). In fact, while example 2 of the Madron problem (Gala and Bagajewicz[26]) requires only 37 s to explore 937 nodes in the cutsets-based tree search, in this example it takes roughly one hour to explore every 1000 nodes in the equations-based tree search without decomposition (All Equations method). Thus, reducing the tree depth by using a decomposition technique is *very beneficial* for nonlinear systems. Although capable of finding the optimal solution, the All Equations method (equation-based without decomposition) is far less computationally efficient than the Decomposed Equations method or even the simpler All Variables method.

Tables 5 and 9 show that the computation time of the All Equations increases from less than one minute in the 13-variable CSTR example to several hours or almost 2 days in the 24-variable mineral flotation process example. We conclude that the All Equations method is severely affected by the scaling problem, which is due to the fact that the number of equations generated (the tree size) usually increases combinatorially with the size of the problem. The number of equations generated in nonlinear systems is also much larger than the number obtained in the same sized linear systems. The main reasons are

(i) Any combination of original equations may result in multiple new equations (not just only one as in linear systems).

(ii) The possibility that there is common variable(s) between a pair of equations is high because one variable (such as the reactor temperature or the flowrate variables) can appear in several equations (instead of at most 2 in linear systems) as can be seen above (recall that the ring sum operation on a pair of cutsets or variables substitution on a pair of equations can be performed only when there exists common stream(s) or common variable(s) between them).

For the mineral process example, the two best methods are the Decomposed Equations method and the Inverted All Variables method. When low or moderate specification is used (MFP1, MFP2), the Decomposed Equations method is the most efficient: it can find the optimal solution within less than one minute. When high specification is used (MFP3), the Inverted All Variables method is the best because this method is tailored for such a design case: in the design case MFP3, the number of sensors in feasible nodes is at least 15; hence, the All Variables method (forward tree search) needs to explore at least 15 levels before it stops, while the inverted tree search explores not more than 9 (=24 minus 15) levels before it stops. This explains the remarkable improvement in computation time by using the inverted tree search.

7.3. Tennessee Eastman Process. Consider the well-known challenge problem, the TE process, which is given in Figure 10 (following the TE process flow diagram shown in Downs and Vogel[33]). The simplified TE model described by Ricker and Lee[32] is used. The steady state operation conditions are generated from the Fortran file that implements the TE model

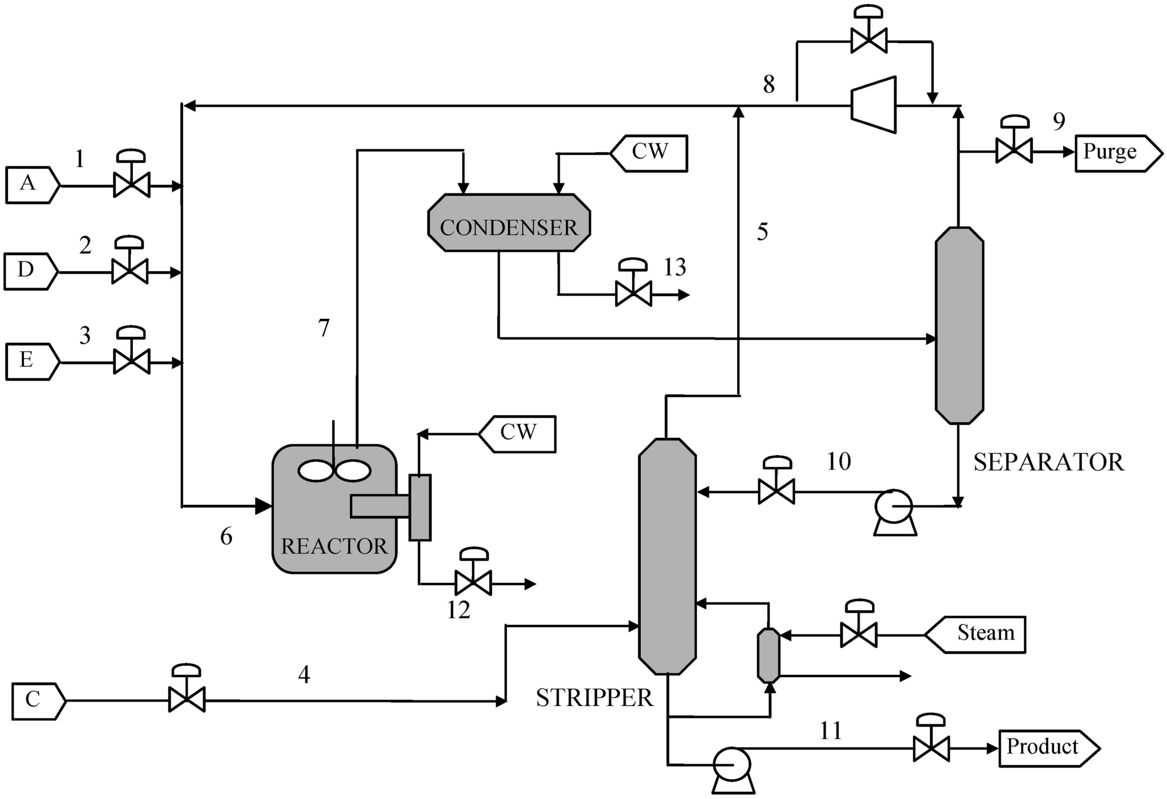

Figure 10. Tennessee Eastman process (following Downs and Vogel[33]).

Table 10. Data for the Tennessee Eastman Problem

variables	nominal operating condition	sensor cost	variables	nominal operating condition	sensor cost
F_6	1889.9	300	$y_{E,8}$	0.186	740
F_7	1475.2	300	$y_{F,8}$	0.023	730
F_{10}	258.56	200	$y_{G,8}$	0.048	740
F_{11}	211.3	200	$y_{H,8}$	0.023	750
$y_{A,6}$	0.322	770	$y_{A,9}$	0.33	720
$y_{B,6}$	0.089	780	$y_{B,9}$	0.138	730
$y_{C,6}$	0.264	730	$y_{C,9}$	0.24	740
$y_{D,6}$	0.069	740	$y_{D,9}$	0.013	750
$y_{E,6}$	0.187	750	$y_{E,9}$	0.186	760
$y_{F,6}$	0.016	760	$y_{F,9}$	0.023	770
$y_{G,6}$	0.035	810	$y_{G,9}$	0.048	780
$y_{H,6}$	0.017	820	$y_{H,9}$	0.023	790
$y_{A,7}$	0.272	750	$y_{D,10}$	0.002	700
$y_{B,7}$	0.114	760	$y_{E,10}$	0.136	710
$y_{C,7}$	0.198	700	$y_{F,10}$	0.016	720
$y_{D,7}$	0.011	710	$y_{G,10}$	0.472	720
$y_{E,7}$	0.177	720	$y_{H,10}$	0.373	730
$y_{F,7}$	0.022	730	$y_{G,11}$	0.537	730
$y_{G,7}$	0.123	780	$y_{H,11}$	0.438	740
$y_{H,7}$	0.084	790	P_r	2806	100
$y_{A,8}$	0.33	780	T_r	393.6	500
$y_{B,8}$	0.138	770	P_s	2734.7	100
$y_{C,8}$	0.24	760	T_s	353.3	500
$y_{D,8}$	0.013	750			

available at Ricker's Web site (http://depts.washington.edu/control/LARRY/TE/download.html). The steady state equations used are

$$y_{i,6}F_6 - y_{i,7}F_7 + \sum_{j=1}^{3} \nu_{ij}R_j = 0 \qquad i = \mathrm{A, B, ..., H} \quad (33)$$

$$y_{i,7}F_7 - y_{i,8}(F_8 + F_9) - x_{i,10}F_{10} = 0 \qquad i = \mathrm{A, B, ..., H} \quad (34)$$

$$z_{i,1}F_1 + z_{i,2}F_2 + z_{i,3}F_3 + F_{i,5} + y_{i,8}F_8 + F_i^* - y_{i,6}F_6 = 0 \qquad i = \mathrm{A, B, ..., H} \quad (35)$$

$$(1 - \varphi_i)x_{i,10}F_{10} - x_{i,11}F_{11} \qquad i = \mathrm{G, H} \quad (36)$$

where ϕ_i (i = G, H) is separation factor of component i in the stripper; $z_{i,j}$, $y_{i,j}$, and $x_{i,j}$ are the molar fractions of chemical i in stream j, which can be feed stream ($z_{i,j}$), liquid stream ($x_{i,j}$), or gas stream ($y_{i,j}$); and ν_{ij} is the stoichiometry factor of chemical i in reaction j. The reaction rates R_j are given by the following expressions:

$$R_1 = \beta_1 V_{\mathrm{V,r}} \exp\left[44.06 - \frac{42600}{RT_{\mathrm{r}}}\right] P_{\mathrm{A,r}}^{\ 1.08} P_{\mathrm{C,r}}^{\ 0.311} P_{\mathrm{D,r}}^{\ 0.874} \quad (37)$$

$$R_2 = \beta_2 V_{\mathrm{V,r}} \exp\left[10.27 - \frac{19500}{RT_{\mathrm{r}}}\right] P_{\mathrm{A,r}}^{\ 1.15} P_{\mathrm{C,r}}^{\ 0.370} P_{\mathrm{D,r}}^{\ 1.00} \quad (38)$$

$$R_3 = \beta_3 V_{\mathrm{V,r}} \exp\left[59.50 - \frac{59500}{RT_{\mathrm{r}}}\right] P_{\mathrm{A,r}}(0.77P_{\mathrm{D},r} + P_{\mathrm{E},r}) \quad (39)$$

where β_j is "tuning" factor of reaction j; $V_{\mathrm{V},r}$ is the liquid volume in the reactor; T_{r} is the temperature in the reactor; and $P_{i,\mathrm{r}}$ is the partial pressure of chemical i in the reactor.

The variables, their nominal operating conditions, and the costs of associated sensors are given in Table 10. Sensor precision of 2% (for all variables) is used.

Values of flowrates F_i are given in kmol/h; P_{r} and P_{s} are the pressures in the reactor and separator, respectively (KPa); T_{r} and T_{s} are the temperatures in the reactor and separator, respectively (K); subscripts A, B, C, D, E, F, G, and H denote components; and subscripts 6, 7, 8, 9, 10, and 11 denote stream number. The variables listed in Table 7 are considered candi-

Table 11. Design Case Studies for the TE Process Example

Design case	TE1 (low spec.)	TE2 (moderate spec.)	TE3 (high spec.)
no. of key variables	6	17	39
key variables	F_6, y_{A6}, y_{G6}, y_{H6}, F_7, F_{10}	F_6, y_{A6}, y_{G6}, y_{H6}, F_7, y_{G7}, y_{H7}, y_{A9}, y_{G9}, y_{H9}, F_{11}, y_{G11}, y_{H11}, P_r, T_r, P_s, T_s	all variables *except* {y_{D9}, y_{E9}, y_{F9}, y_{G9}, y_{H9}, F_{10}, y_{D10}, y_{E10}}
requirement	observability	observability	redundancy
precision thresholds	2%	2%	1.5% 1.6% (y_{G8}, y_{H8})
residual precision thresholds			4% for *all key variables except* {y_{G8}, y_{H8}, P_r, T_r, P_s, T_s}
measured variables	F_6, y_{A6}, y_{G6}, y_{H6}, F_7, F_{10}	F_6, y_{A6}, y_{G6}, y_{H6}, F_7, y_{A7}, y_{A9}, y_{G9}, y_{H9}, y_{G10}, y_{H10}, F_{11}, y_{H11}, P_r, T_r, P_s, T_s	*all variables except* {y_{E9}, F_{10}, y_{E10}, P_r} (43 variables in total)
sensor cost	3200	9630	29 640

Table 12. Results for the TE Process Example

case study		TE1	TE2	TE3
All Equations method		not used		
decomposed eqs (9 subgraphs)	computation time	2 min, 40 s	>74 h (all current best solutions obtained in 14 h)	>10 h (suboptimal solution found in one second)
	computation time to generate equations	<1 s	<1 s	<1 s
	number of equations generated	61	67	68
	number of nodes explored	11 628	>1.8 million (all "current best" solutions obtained within first 510 000 nodes explored)	>500 000 (suboptimal solution found at node 44)
All Variables	computation time	9 h, 8 min	>3 days (45 days estimated)	not used
	number of nodes explored	1 867 295	>6 million	
Inverted All Variables	computation time	not used	not used	4 min, 16 s
	number of nodes explored			1726

Table 13. Results Used for Selecting the Right Tree Search Strategy

	design case		
	MFP1	MFP2	MFP3
computation time (s)	2	16	1
average number of sensors in feasible solution	16.45	16.55	20.36
number of sensors in optimal solution	10	12	15

dates for measurements; other variables in the TE process (e.g., input flowrates F_1, F_2, F_3) are assumed to be either known by measurements (forced measurements) or of little importance for consideration. The total number of equations involving listed variables is 28.

Three design cases are considered, and they are shown in Table 11.

For this example, the All Equations method was not used because (i) the computation time to generate all balance equations from ($2^{28} - 1$) combinations of 28 original equation is too long and (ii) its large tree size (large number of equations) leads to a long computation time. The computation time and number of nodes explored for the other methods are given in Table 12.

For the case studies with low and moderate specification (TE1, T2), the Inverted All Variable method is not used because the inverted tree search will perform poorly for the design case with low specification like TE1. The case study TE2 is similar to the case study MFP1 (mineral flotation example) where the number of sensors in the optimal (or suboptimal) solution accounts for less than a half of all sensors; hence, the performance of the inverted tree search is predicted to be comparable to that of the All Variables method, which performs poorly as shown in Table 12.

The proposed methods can find optimal solution for the two design cases with low and high specification (TE1 and TE3, respectively) with the Decomposed Equations method being the best method for the design case TE1 while the Inverted All Variables method is the best method for TE3 as expected. Optimal solutions for these two design cases are found within less than 5 min.

For design case with moderate specification (TE2), the Decomposed Equations method is the only viable option when the factor of acceptable computation time is desired. In fact, after roughly 3 days of running time and 6 million nodes explored, the "current best" solution found by the All Variables method consists of 21 sensors with the cost of 14 120, far worse than the solution found by the Decomposed Equations method. We assume that the computation time of the All Variables method is comparable to the computation time of the same sized linear system, the CDU process presented in ref 27, which was estimated to be 45 days. The Decomposed Equations method was then used: the number of decompositions is 8 (original graph is decomposed into 9 subgraphs), and the number of equations generated is 67. All the "current best" solutions (the incumbent) were found within the first 510 000 nodes explored, 14 h computation time. After that the tree search procedure kept running for 60 h and explored a further 1.3 millions nodes without finding any better solution before it was terminated (so in total, 74 h running time and 1.8 million nodes explored). The current best solution is reported here, which is to measure F_6, y_{A6}, y_{G6}, y_{H6}, F_7, y_{A7}, y_{A9}, y_{G9}, y_{H9}, y_{G10}, y_{H10}, F_{11}, y_{H11}, P_r, T_r, P_s, and T_s. It has a cost of 9630. This solution contains only 14 of the 17 key variables: F_6, y_{A6}, y_{G6}, y_{H6}, F_7, y_{A9}, y_{G9}, y_{H9}, F_{11}, y_{H11}, P_r, T_r, P_s, and T_s.

Finally, case study TE3 is one that will require a lot of measurements, because the requirements are very strict. In this case, the best solution obtained by the Decomposed Equations after roughly 500 000 nodes explored and 10 h of computation time (before it was terminated) is to measure all variables except three variables (y_{E9}, F_{10}, y_{E10}) with a cost of 29 740. This solution contains one more sensor for measuring P_r and a cost slightly higher (100 more) when compared with the optimal solution found by the inverted tree search. Thus, even in the case that the Decomposed Equations method has to be terminated when the computation time becomes unacceptably long (e.g., with large scale problems with high number of key variables) such that the finding of an optimal solution is not guaranteed, the method finds suboptimal solutions very near the global optimal solution within an acceptable time (in fact, this suboptimal solution is found at node 44, just after 1 s running time).

The results shown here point out that (i) when the level of desired properties (the specifications) of sensor network is either low or high (e.g., either a small or a large number of key variables is involved), even a large scale nonlinear sensor network problem like the TE problem can be solved efficiently using either the Decomposed Equations method (for a low level of specifications) or inverted tree search method (for a high level of specifications). (ii) For realistic design cases of large scale nonlinear problem where the level of desired properties is neither low nor high, the Decomposed Equations method is able to find optimal or near-optimal solution within an acceptable time; however, the optimality is not guaranteed because the computation process has to be terminated halfway.

A more computationally efficient method for designing a realistic large scale nonlinear sensor network is left for future work.

8. Choice of Strategy

It can be seen that the inverted tree search remarkably improves the computational time when a high level of specifications is desired. A step further is to intelligently select which strategy to use corresponding to a specific design case to have the shortest computational time. The criterion of when to choose the inverted tree search instead of a forward tree search, inferred empirically by our observations of testing problems, is to choose the inverted tree search when the followings conditions are met:

(i) The number of variables is more than 15

(ii) The average number of sensors in feasible solutions is more than 70% of the total number of variables (based on the first 10 feasible solutions found).

The first condition is needed because (a) the equations-based methods (forward strategy) can solve problems involving not more than 15 variables very efficiently, and hence, inverted tree search is not needed for this type of small scale problem, and (b) if number of variables is less than 15, usually the computation time is short and the time to compute the average number of sensors in feasible solutions offsets the gain in computation time (if any) by using the inverted tree search.

The procedure to quickly calculate the average number of sensors in feasible solutions that is as close as possible to the number of sensors in the optimal solution has three steps:

(i) Prepare two lists of variables: one list of key variables (list one) and a list of nonkey variables (list two).

(ii) If measuring *all* key variables is a feasible solution, then use the number of key variables as final result and stop; otherwise, go to step iii.

(iii) Employ the tree enumeration procedure using nonkey variables from list two to find the first 10 combinations of *all* key variables (list one) and nonkey variables (from list two) that are feasible solutions.

The average number of sensors in feasible solutions and the associated computation time for the four mineral process design cases are shown in Table 13.

According to the results shown in Table 13 and the criteria presented above, the design case MFP3 should be solved by using an inverted tree search. The calculation results shown in Table 9 confirm that this is the "right" choice. The two design cases MFP1 and MFP2 should be solved by using the Decomposed Equations method (forward strategy).

9. Conclusions

In this paper, the equations-based tree search method for the design of nonlinear sensor network was presented. The proposed method is guaranteed to find an optimal solution and is computationally efficient for small scale and middle scale problems. However, its performance is not always satisfactory when dealing with large scale problems. Another version of the tree search method, the inverted tree search using the list of variables, was also presented. The inverted strategy is tailored for design cases with a high level of specifications and is shown to remarkably improve the computation time, especially with large scale nonlinear problems like the TE process where it solved a high specifications design case within a few minutes.

Nomenclature

c_i = cost of sensor i

q_i = binary variable indication of whether sensor i is used

σ_i = precision of estimator i

σ_i^* = precision threshold of estimator i

F_i = flowrate of stream i

h_i = enthalpy of stream i

c_{ij} = concentration of component j in stream i

δ_i = stoichiometric coefficient of component i in chemical reaction

T_r, P_r = temperature and pressure of reactor

T_s, P_s = temperature and pressure of separator

$P_{i,r}$ = partial pressure of chemical i in the reactor

T_f = flash drum temperature

F_i, T_i, c_{Ai} = inlet flowrate, temperature, and concentration of A of the CSTR

F, T, c_A = outlet flowrate, temperature, and concentration of A of the CSTR

F_{ci}, T_{ci} = inlet flowrate and temperature of coolant in the CSTR

F_c, T_c = outlet flowrate and temperature of coolant in the CSTR

F_{vg} = flowrate of vent gas leaving the CSTR

U, A = heat transfer coefficient and heat transfer area in the CSTR

C_d = catalyst activity

V_j = volume of jacket

V = reactor volume

$V_{v,r}$ = liquid volume in the reactor

C_p, ρ = heat capacity and density of fluid mixture in the CSTR

C_{pj}, ρ_j = heat capacity and density of coolant

K_i = vapor−liquid equilibrium ratio of component i

E = activation energy of chemical reaction

k_0 = pre-exponential factor or frequency factor

α = reaction order

ν_{ij} = stoichiometry factor of chemical i in reaction j

ϕ_i = separation factor of component i in the stripper

β_j = "tuning" factor of reaction j

R = universal gas constant

R_j = reaction rate of reaction j

ΔH_{rxn} (or simply ΔH) = heat of reaction

Q_{vap} = amount of heat removed in the flashing process

$z_{i,j}$, $x_{i,j}$, and $y_{i,j}$ = molar fraction of component i in stream j, which can be feed stream ($z_{i,j}$), liquid stream ($x_{i,j}$), or gas stream ($y_{i,j}$)

Literature Cited

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

(2) Kretsovalis, A.; Mah, R. S. H. Effect of Redundancy on Estimation Accuracy in Process Data Reconciliation. *Chem. Eng. Sci.* **1987**, *42*, 2115–2121.

(3) Madron, F.; Veverka, V. Optimal Selection of Measuring Points in Complex Plants by Linear Models. *AIChE J.* **1992**, *38* (2), 227–236.

(4) Ali, Y.; Narasimhan, S. Sensor Network Design for Maximizing Reliability of Linear Processes. *AIChE J.* **1993**, *39* (5), 820–828.

(5) Raghuraj, R.; Bhushan, M.; Rengaswamy, R. Locating Sensors in Complex Chemical Plants Based on Fault Diagnostic Observability Criteria. *AIChE J.* **1999**, *45* (2), 310–322.

(6) Bhushan, M.; Rengaswamy, R. Design of Sensor Network Based on the Signed Directed Graph of the Process for Efficient Fault Diagnosis. *Ind. Eng. Chem. Res.* **2000**, *39*, 999–1019.

(7) Bhushan, M.; Rengaswamy, R. Comprehensive Design of Sensor Networks for Chemical Plants Based on Various Diagnosability and Reliabilty Criteria. I. Framework. *Ind. Eng. Chem. Res.* **2002**, *41*, 1826–1839.

(8) Bhushan, M.; Rengaswamy, R. Comprehensive Design of Sensor Networks for Chemical Plants Based on Various Diagnosability and Reliabilty Criteria. II. Applications. *Ind. Eng. Chem. Res.* **2002**, *41*, 1840–1860.

(9) Musulin, E.; Bagajewicz, M.; Nougues, J. M.; Puigjaner, L. Instrumentation Design and Upgrade for Principal Components Analysis Monitoring. *Ind. Eng. Chem. Res.* **2004**, *43*, 2150–2159.

(10) Bagajewicz, M.; Fuxman, A.; Uribe, A. Instrumentation Network Design and Upgrade for Process Monitoring and Fault Detection. *AIChE J.* **2004**, *50* (8), 1870–1880.

(11) Meyer, M.; Le Lann, J.; Koehret, B.; Enjalbert, M. Optimal Selection of Sensor Location on a Complex Plant Using a Graph Oriented Approach. *Comput. Chem. Eng.* **1994**, *18* (Suppl), S535−S540.

(12) Luong, M.; Maquin, D.; Huynh, C. Ragot, J. Observability, Redundancy, Reliability and Integrated Design of Measurement Systems. *Proceedings of 2nd IFAC Symposium on Intelligent Components and Instrument Control Applications*, SICICA '94, Budapest, Hungary, June 8−10, 1994. (Accessed via the Internet at http://citeseer.ist.psu.edu/luong94observability.html.)

(13) Bagajewicz, M. Design and Retrofit of Sensors Networks in Process Plants. *AIChE J.* **1997**, *43* (9), 2300–2306.

(14) Bagajewicz, M.; Sanchez, M. Reallocation and Upgrade of Instrumentation in Process Plants. *Comput. Chem. Eng.* **2000**, *24*, 1945–1959.

(15) Bagajewicz, M. *Design and Upgrade of Process Plant Instrumentation*; Technomic Publishers: Lancaster, PA, 2000.

(16) Chmielewski, D.; Palmer, T.; Manousiouthakis, V. On the Theory of Optimal Sensor Placement. *AIChE J.* **2002**, *48* (5), 1001–1012.

(17) Bagajewicz, M.; Cabrera, E. A New MILP Formulation for Instrumentation Network Design and Upgrade. *AIChE J.* **2002**, *48* (10), 2271–2282.

(18) Bagajewicz, M.; Cabrera, E. Pareto Optimal Solutions Visualization Techniques for Multiobjective Design and Upgrade of Instrumentation Networks. *Ind. Eng. Chem. Res.* **2003**, *42*, 5195–5203.

(19) Sen, S.; Narasimhan, S.; Deb, K. Sensor Network Design of Linear Processes Using Genetic Algorithms. *Comput. Chem. Eng.* **1998**, *22*, 385–390.

(20) Carnero, M.; Hernandez, J.; Sanchez, M.; Bandoni, A. An Evolutionary Approach for the Design of Nonredundant Sensor Networks. *Ind. Eng. Chem. Res.* **2001**, *40*, 5578–5584.

(21) Carnero, M.; Hernandez, J.; Sanchez, M.; Bandoni, A. On the Solution of the Instrumentation Selection Problem. *Ind. Eng. Chem. Res.* **2005**, *44*, 358–367.

(22) Heyen, G., Dumont, M.; Kalitventzeff, B. Computer-aided design of redundant sensor networks. In *Proceedings of 12th European Symposium on Computer-aided Process Engineering*; Grievink, J., van Schijndel, J., Eds.; Elsevier Science: Amsterdam, 2002; pp 685−690.

(23) Singh, A. K.; Hahn, J. Determining Optimal Sensor Locations for State and Parameter Estimation for Stable Nonlinear Systems. *Ind. Eng. Chem. Res.* **2005**, *44*, 5645–5659.

(24) Singh, A. K.; Hahn, J. Sensor Location for Stable Nonlinear Dynamic Systems: Multiple Sensor Case. *Ind. Eng. Chem. Res.* **2006**, *45*, 3615–3623.

(25) Muske, K. R.; Georgakis, K. Optimal Measurement System Design for Chemical Processes. *AIChE J.* **2003**, *49* (6), 1488–1494.

(26) Gala, M.; Bagajewicz, M. J. Rigorous Methodology for the Design and Upgrade of Sensor Networks Using Cutsets. *Ind. Eng. Chem. Res.* **2006**, *45* (20), 6687–6697.

(27) Gala, M.; Bagajewicz, M. J. Efficient Procedure for the Design and Upgrade of Sensor Networks Using Cutsets and Rigorous Decomposition. *Ind. Eng. Chem. Res.* **2006**, *45* (20), 6679–6686.

(28) Ali, Y.; Narasimhan, S. Sensor Network Design for Maximizing Reliability of Bilinear Processes. *AIChE J.* **1996**, *42* (9), 2563–2575.

(29) Mah, R. S. H. *Chemical Process Structures and Information Flows*; Butterworths: Stoneham, MA, 1990.

(30) Smith, H. W.; Ichiyen, N. Computer Adjustment of Metallurgical Balances. *Can. Znst. Mining Metall. (C.Z.M.) Bull.* **1973**, *66*, 97–100.

(31) Narasimhan, S.; Jordache, C. *Data Reconciliation & Gross Error Detection: An Intelligent Use of Process Data*; Gulf Publishing: Houston, TX, 2000.

(32) Ricker, N. L.; Lee, J. H. Nonlinear Modeling and State Estimation for the Tennessee Eastman Challenge Process. *Comput. Chem. Eng.* **1995**, *19* (9), 983–1005.

(33) Downs, J. J.; Vogel, E. F. A Plant-wise Industrial Process Control Problem. *Comput. Chem. Eng.* **1993**, *17* (3), 245–255.

Received for review June 26, 2007
Revised manuscript received February 25, 2008
Accepted April 1, 2008

IE0708778