New Efficient Breadth-First/Level Traversal Tree Search Method for the Design and Upgrade of Sensor Networks

Duy Quang Nguyen and Miguel J. Bagajewicz
School of Chemical, Biological and Materials Engineering, The University of Oklahoma, Norman, OK 73019

DOI 10.1002/aic.12332
Published online August 31, 2010 in Wiley Online Library (wileyonlinelibrary.com).

This work studies the problem of optimally locating sensors for monitoring chemical processes, formally known as the sensor network design and upgrade problem. This problem is an integer programming problem and has been solved to global optimality only using tree search methods using depth-first strategy. In this article, we exploit certain cost properties of the different nodes in the tree to efficiently prune nonoptimal nodes using a breadth-first/level traversal tree search method to obtain the global optimum. We show that this strategy allows cutting considerably the computational time for problems that the depth-first strategy was inefficient. © 2010 American Institute of Chemical Engineers *AIChE J,* 57: 1302–1309, 2011

Keywords: instrumentation design, instrumentation upgrade

Introduction

The importance of data treatment techniques like data reconciliation and gross error detection in the process industry 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. With use of data treatment techniques and considering the fact that, in reality, only a fraction of process variables are measured by sensors, the process-monitoring capabilities (e.g., observability, redundancy, and biases detectability) of the instrumentation network depends on the location and precision of the sensors. Thus, the problem of optimally and systematically placing sensors in process plants arises naturally and formally known as the sensor (instrumentation) network design problem (SNDP). Research on this problem is extensive; a review of previous works up to year 2000 can be found in Bagajewicz.[1] A few recent relevant works with focus on optimization methods are described next.

The most common SNDP problem formulation is minimizing sensors cost subjected to constraints (specifications) on observability, redundancy, and precision of variables of interest (key variables), that is, the objective is to design cost-optimal sensor network for process-monitoring purposes. This type of SNDP was traditionally solved by using graph methods (as in the works of Kretsovalis and Mah[2] and Meyer et al.[3]) or matrix manipulation techniques (as in Madron and Veverka[4]). The aforementioned techniques have the limitation that they solve the problem to achieve observability of variables.

Bagajewicz[5] was the first to formulate the problem as a mixed integer nonlinear programming problem (MINLP), which was then solved by using a depth-first tree search procedure. Bagajewicz and Cabrera[6] proposed an MILP model and Chmielewski et al.[7] used linear matrix inequalities to obtain convex MINLP formulations. Bagajewicz and Cabrera's[6] method guarantees optimality but it suffers from scaling problems, that is, the computational time becomes unacceptably long when dealing with large-scale problems. In addition, this method as well as the one proposed by Chmilewski et al.[7] can only be used to solve minimum-cost sensor network problems subject to precision constraints when data reconciliation is used to exploit software redundancy. Most recently, Kelly and Zyngier[8] presented an MILP model based on the Schur complements theorem to design sensor network for process-monitoring purpose. The authors showed that the model can

Correspondence concerning this article should be addressed to M. J. Bagajewicz at bagajewicz@ou.edu.

quickly find "good" solutions but locating global optimum solution is too time consuming. The methods were never extended to include other constraints, like accuracy, reliability, etc., and it is unclear if that is possible.

Recent developments include the use of a genetic algorithm, which was first used to solve an SNDP by Sen et al.[9] Carnero et al.,[10,11] in turn, used the genetic algorithms to solve multiobjective design problems. The genetic algorithms seem to perform well but they do not guarantee optimality.

The depth-first tree search method presented by Bagajewicz[5] was recently improved by Gala and Bagajewicz[12,13] and Nguyen and Bagajewicz.[14] Gala and Bagajewicz[12] explored combinations of cutsets (instead of individual measurements) using a tree enumeration search, which was then enhanced by using a graph decomposition technique (Gala and Bagajewicz[13]). The cutset-based depth-first methods are computationally efficient but applicable to linear systems only. Later, Nguyen and Bagajewicz[14] extended the cutset-based methods to solve nonlinear problems by replacing cutsets with process model equations; they also proposed a special tree search method tailored for problems with high level of specifications, called the inverted tree search. Although very efficient in certain cases, both the equation-based method and the inverted tree search fail to perform well in realistic large-scale nonlinear problems (Nguyen and Bagajewicz[14]).

Although Bagajewicz and Sanchez[15] developed a method to achieve observability and Bagajewicz and Sanchez[16] developed a method for reliable sensor networks, it was the cost-optimal formulation,[5] that prevailed as the main paradigm. Many constraints were added later: reallocation was considered,[17] and the impact of maintenance was also studied.[18] The duality with the cost constrained maximum precision was also established.[19] All these problems used the depth-first search methods, which are to this date the only methods that guarantee global optimality when constraints other than precision are used.

In this work, we aim at developing a new computational method to solve the cost-optimal SNDP for process-monitoring purpose of chemical processes, especially, the nonlinear problems where previously developed depth-first tree search methods show limitations. We propose a new tree search method: instead of exploring the tree through its branches, i.e., adding one instrument at a time and pruning branches using a certain stopping criteria, we resort to looking at configurations with the same number of instruments, that is, looking at all branches at the same level of the tree. The method we proposed is known as a breadth-first strategy (Diwekar[20]). In addition, we propose a few modifications to these strategies, and we also provide stopping criteria that are particular to minimum-cost problems like the one we intend to solve. The method we propose belongs to the breadth-first strategy category (in the sense that it expands first all successor/sister nodes of the current node rather than going down the tree) but it is not exactly the breadth-first branch and bound method as described in optimization textbooks, so we name it "level traversal" tree search.

This article is structured as follows: we first review the problem definition, and we follow with a brief review of the existing tree search techniques. We then present our level traversal strategy (breadth-first tree search) and we finish with illustrations.

Problem Formulation

The mathematical model to design minimum-cost sensor network for process-monitoring purpose (in its simplest form, Bagajewicz[5]) is as follows

$$\left.\begin{aligned}
&\text{Min} \sum_{\forall i} c_i q_i \\
&s.t. \\
&\sigma_i(q) \le \sigma_i^* \qquad \forall i \in M_\text{S} \\
&q_i = 0, 1 \qquad \forall i
\end{aligned}\right\} \tag{1}$$

where q_i is the vector of binary variable indicating that a sensor is located in variable i, c_i is the cost of such a sensor, M_s represents the set of variables where a certain specification is required (desired level of precision/residual precision or error delectability, etc.), $\sigma_i(q)$ is the value of the property under consideration (e.g., precision) corresponding to the set of sensors represented by q and σ_i^*, the corresponding threshold value.

This problem is a MINLP (the vector q consists of binaries while the constraints $\sigma_i(q) \le \sigma_i^*$ are nonlinear).

The data reconciliation problem is an inherent part of the SNDP. Indeed, the data reconciliation problem needs to be solved to calculate the precision/residual precision, etc. of key variables, so as to determine whether a candidate solution satisfies the required specifications. Solving data reconciliation problem of partly measured systems is not straightforward. Chmielewski et al.[7] showed that an unmeasured variable can be modeled in data reconciliation formulation using a fake sensor with very high variance. This approach greatly facilitates the solving of data reconciliation and is utilized in this work. Note that, as described in Nguyen and Bagajewicz,[14] for nonlinear problems, the process model equations are first linearized around nominal operating points before the data reconciliation problem can be solved. Sensitivity analysis of an illustrated example presented in Nguyen and Bagajewicz[14] shows that, for nonlinear problems, the obtained optimal network is valid within the normal variation range of process variables (about 30% of the nominal values), but it is not valid for a wider range of process operating conditions. The illustrated example used in this work is a nonlinear problem. Other problems extend the constraints of (1) to accuracy and other constraints.[5]

Tree Search Methods

Various methods have been used to solve the problem (1). Previous approaches to solve the problem can be classified into two groups:

- Transforming the problem into well-established optimization problems (MILP, convex optimization using linear matrix inequalities techniques) by introducing auxiliary variables, which then can be solved by using available commercial software packages such as GAMS or Matlab. It works well for small-scale linear systems and for precision constraints only.
- Using a tree search method. The base unit can be single measurement (Bagajewicz[5]) or cutset of process graph as base unit (Gala and Bagajewicz[12,13]).

The direct enumeration tree search method is illustrated in Figure 1. The procedure is as follows:

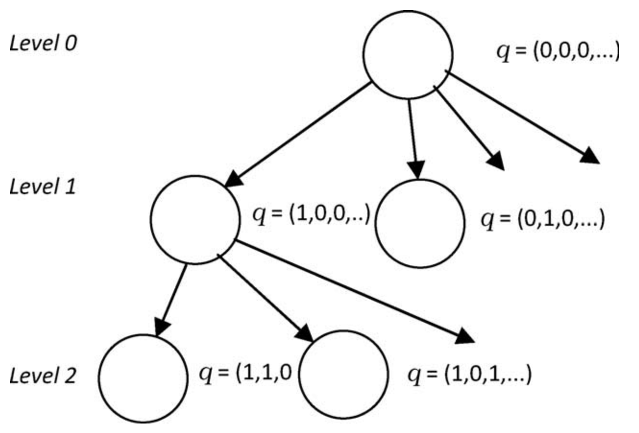


Figure 1. Tree search method.

- Start with a root node with no variables being measured ($q = 0$), it is trivially infeasible.
- Develop each branch and making one element of q active until the stopping criterion is met. Then, back up one level and develop next branch using a branching criteria.

Operations in each node

Each node (a specific value of vector q) is checked for cost and feasibility. The cost is $\sum_{\forall i} c_i q_i$, which is the sum of costs of all the selected sensors. The data reconciliation problem is solved to calculate/determine the precision, redundancy, etc. of key variables, on which the feasibility of the node can be determined.

Branching criteria

Sensors are added to nodes in the direction of minimum cost, that is, the sensor chosen to be added to the current node has the cheapest cost among all candidates.

Stopping criteria

In each node, the following two operations are performed in sequential order to determine if we need to continue exploring the current branch: (i) stop if the cost of the current node is more than the current best because even if the node is feasible, it cannot compete with the current best, (ii) stop if the node is feasible; update the current best if the cost of this feasible node is less than the current best.

This stopping criterion is valid for both depth-first tree search and level traversal (breadth-first) tree search (described below). In depth-first tree search, if the stopping criterion is met, one should stop, back up one level, and develop the next branch, as any node below will be more expensive.

The depth-first tree search method is not efficient for medium- and large-scale problems because computational time increases exponentially with the size of the problem. To deal with these problems, tree search methods based on cutsets were proposed (Gala and Bagajewicz[12,13]). The cutsets-based methods were based on the notion that a cutset corresponds to a set of variables with which a material balance can be written. This important characteristic of cutset was utilized by Gala and Bagajewicz[12] who replaced individual measurement by cutset as base unit in the tree search algorithm. The tree search procedure is essentially the same as the individual measurement-based procedure described above except that:

(i) A preprocessing step is needed before the tree search can be conducted, which is to find all cutsets containing at least one variable of interest (called key variable) of the problem.

(ii) In each node, the solution (to be evaluated for cost and feasibility) is the union of active (selected) cutsets.

(iii) When the stopping criterion is satisfied, one should back up two levels.

The virtue of this cutsets-based method is that only meaningful measurements that contribute to the redundancy or observability of variables of interest are added through the use of cutsets. Moreover, when adding a cutset, several measurements may be added at a time instead of only one as in single measurement-based tree search. These two properties help cutsets-based tree search to find feasible nodes much more rapidly than the single measurement-based counterpart.

Although tree enumeration using cutsets is suitable for middle-scale problem (number of streams ≥ 20), it still has one limitation, which is that for large-scale problems (number of streams ≥ 40), the number of cutsets may be too large, and the number of nodes in the tree that needs to be explored is prohibitively large: hence, the computation time can be as long as several days. To overcome this computational limitation, Gala and Bagajewicz[13] proposed the "decomposition of process graph network" algorithm. The algorithm still makes use of cutsets, but the process graph is decomposed into several subgraphs to reduce the number of cutsets in the candidate lists and hence, reduce the size of the tree. Original cutsets that contain elements of different subgraphs are reconstructed when two cutsets from these subgraphs are picked. The tree search procedure is almost the same as the procedure without decomposition except that the branching and stopping criterion are modified, and ring-sum operations between active cutsets are performed to find the cutsets that are "missed" when compared with the cutsets list of original process graph. This cutsets-based tree search coupled with decomposition technique has been shown to be a very efficient method for solving linear large-scale problems (Gala and Bagajewicz[13]). Later, Nguyen and Bagajewicz[14] extended the cutset-based method to solve nonlinear problems by using process balance equations and incidence matrix manipulation instead of cutsets and graph decomposition (called equations-based method).

Nguyen and Bagajewicz[14] also presented a technique based on an inverted tree search. 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). This method is very efficient for problems with high level of specifications (e.g., when there are many key variables or redundancy is required) where feasible solutions contain a large portion of available sensors. Table 1 summarizes the most suitable methods (that were developed in our group) for each case.

Note that in Table 1, the cutsets-based and equations-based methods are meant to be the ones with decomposition (which is always better than the corresponding versions

Table 1. **Most Suitable Method for Solving Sensor Network Design Problem**

Level of Specifications	Linear Systems	Nonlinear Systems
Low	Cutsets-based or measurement-based tree search	Equations-based or measurement-based tree search
Medium	Cutsets-based	Equations-based
High	Cutsets-based or inverted tree search	Equations-based or inverted tree search

without decomposition). It can be noted that the cutsets-based and equations-based methods (with decomposition) are the best choice for all levels of specifications while occasionally, these methods are outperformed by the measurement-based tree search or inverted tree search for problems with low level and high level of specifications, respectively.

We now present a level traversal (breadth-first) technique which takes advantage of certain properties of trees that are constructed using the minimum-cost branching criteria.

New Methodology—Level Traversal Search

We start first by defining the following terms:

Sister nodes: a given node is a sister node of a current node when: (i) it is at the same level (same number of sensors) and on the right of the current node), (ii) both share the same parent node.

Families of nodes: a set (family) of nodes that have the same number of sensors (same number of active elements) and share the same root (same parent).

Head of family: the leftmost node in the family of nodes, that is, the cheapest node.

To illustrate the above concepts, consider a list of sensors in ascending order of cost: 123456. At level three, the following nodes (consisting of three sensors) (123, 124, 125, 126) are said to form a family of nodes with parent root 12 (Figure 2). The next families of nodes at the same level are (134, 135, 136) with parent root 13; (145, 146) with parent root 14, and finally (156) with parent root 15. The heads of families are 123, 134, 145, and 156.

We now note the following properties when the tree is built using the cheapest candidate (minimum-cost branching criteria).

- Property 1: There is cost monotonicity within each family, that is, the cost increases as one moves within each family from left to right.

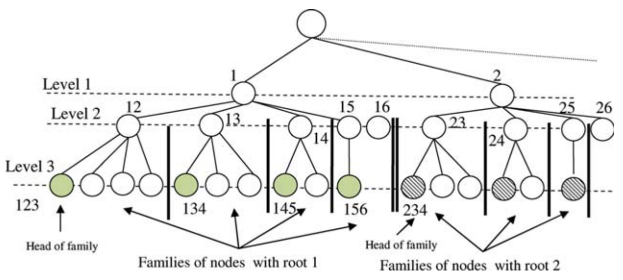
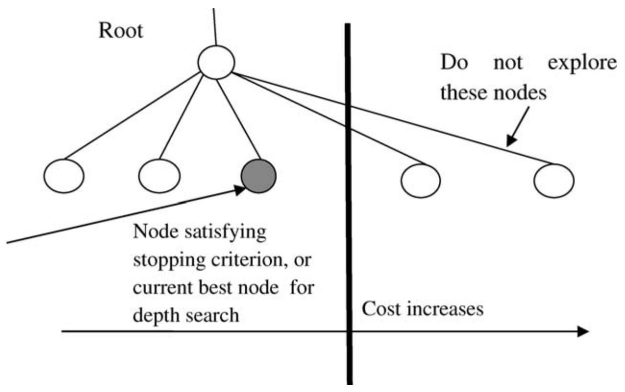


Figure 2. Families of nodes.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]



Root
Do not explore these nodes
Node satisfying stopping criterion, or current best node for depth search
Cost increases

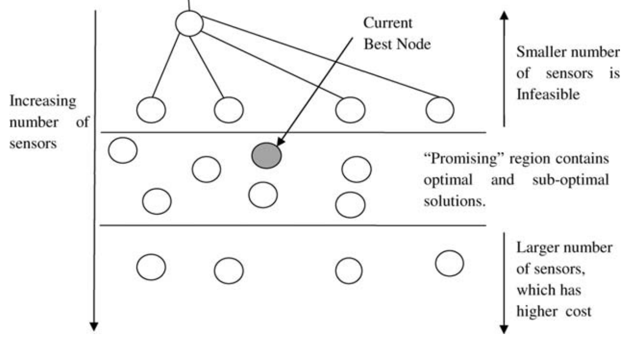
Figure 3. Stopping criterion.

- Property 2: There is cost increasing monotonicity among heads of family that share the same root.

Property 1 is straightforward: it stems from the minimum-cost branching criteria. Property 2 is also self-evident from the branching criteria. For example, node 123 has smaller cost than node 134 and so on. This is because they share the same root (node 1). However, a member of one family can in fact have a larger cost than members of a family on the right. For example, node 125 (third member of the first family) can have higher cost than node 134 (head of the second family).

Properties 1 and 2 can be used efficiently in a level traversal strategy. Suppose that one node at level 3 is found to satisfy the stopping criteria (it is feasible and its cost is smaller than the current upper bound, or simply if it is more costly than the current upper bound). The cost monotonicity within each family and among heads of family implies that if the node satisfying the stopping criteria is a head of family, all nodes that are on the right side and share the same root with that node would have higher cost. In such case, one can directly omit looking at all families sharing the same root and move to the families in the next root. On the other hand, if the node satisfying the stopping criteria is not a head of family, then one simply skips its sister nodes and moves on to the next family. This is illustrated next.

If the head of the first family "123" is found to satisfy the stopping criterion, the next node to explore is node "234," the leftmost node that does not share the same root with node "123." If node "123" does not satisfy the stopping criteria, the tree search continues exploring other nodes



Current Best Node
Increasing number of sensors
Smaller number of sensors is Infeasible
"Promising" region contains optimal and sub-optimal solutions.
Larger number of sensors, which has higher cost

Figure 4. Searched space in the level traversal search.

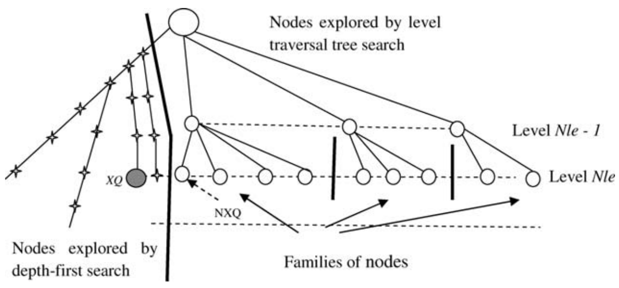

Nodes explored by level traversal tree search
Level Nle - 1
XQ
Level Nle
NXQ
Nodes explored by depth-first search
Families of nodes

Figure 5. Level by level search.

of this family (124, 125, 126). If node “124” satisfies the stopping criteria, its sister nodes (“125” and “126”) can be skipped. Because these three nodes are not head of family, no matter whether they satisfy the stopping criterion or not, the tree search moves on to explore the next family (the second family whose head is “134”). The same procedure is repeated for the second family (e.g., if “134” satisfies the stopping criteria, the next node to explore is node “234”). The tree search would continue in that fashion: it continues looking for nodes that satisfy the stopping criterion and skips nodes that have higher cost than the current best as it marches from left to right until the whole level under investigation is explored.

This discussion points out that:

- The level traversal tree search strategy is very efficient if a current best is identified in left-hand side of the tree, in such case, lots of nodes in the right side of that current best node can be eliminated. For example, if node “1234” is identified to be current best (the leftmost node in the level consisting of four sensors), then no other nodes in this level can compete with the node 1234 and one can quickly move to the upper levels.
- Conversely, if the level traversal tree search cannot identify a node satisfying the stopping criterion until the end (the nodes on the rightmost side) of the current level, the tree search has to explore the whole level (explore all nodes from left to right). If this situation occurs, the level traversal tree search is not efficient to solve large-scale problems. Some sort of “inverted search” based on exploring the tree from right to left would have to be used. This is beyond the scope of this article.

We now proceed to describe the calculation procedure.

We start with a depth-first search using the branching criterion that selects the cheapest sensor. This strategy is not efficient for medium- and large-scale problems, so the depth-first procedure stops when the number of nodes explored reach a predefined limit. Assume that the current best node has been identified.

Because the current best (identified by depth-first search) is unlikely to be global optimum, the tree search continues seeking for global optimum by exploring the nodes in the right-hand side and at the same level with the current best (that is, breadth-first or level traversal strategy). In this strategy, within a family of nodes, tree search looks for a node satisfying stopping criterion and updates the current best if applicable. Because sensors are added in the direction of minimum cost, the sister nodes of this node have higher cost than the current best so they are disregarded (Figure 3). The

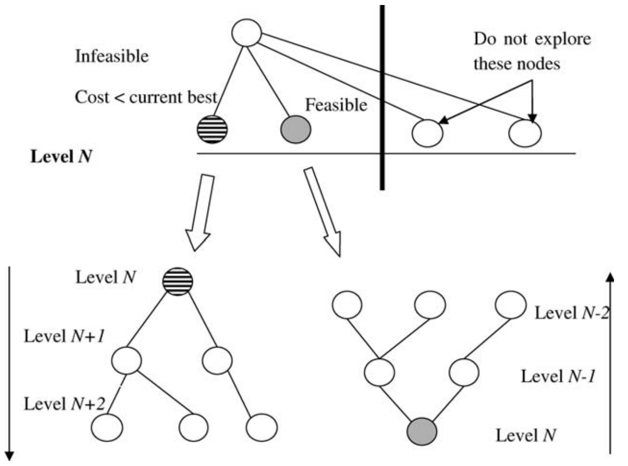


Figure 6. Hybrid vertical and level by level search.

search should continue with the next families of nodes at the same level and families in upper levels until we identify a level where all nodes are infeasible (Figure 4).

Thus, the essence of the new method is to search the tree horizontally within the “promising region” only, defined as some number of levels above the currently identified. Compared with the depth-first tree search method, this method saves computational time by skipping the region above the “promising region” where nodes are infeasible. The question is how to explore this region horizontally and how to guarantee global optimality. For this we present two methods:

Level by level search

The procedure of this method is as follows:

(1) Run the depth-first tree search method. Record the current best solutions and the associated depth level (number of sensors) in those solutions.

(2) Stop if the number of nodes explored reaches the predefined limit. This limit depends on the size of the problem. The current best solution found is denoted as *XQ* and its depth level (the number of sensors) is denoted as *Nle* (see Figure 5). At this point, all nodes in the left of *XQ* and their children have been explored.

(3) If node *XQ* is a leftmost head of a family, identify its parent and *move* to the next level up (*Nle* − 1). If not stay at level *Nle*. Either way go to Step 4.

(4) Identify all the families of nodes at the current level and on the right-*hand* side of *XQ*.

(5) In each family, identifying the node that satisfies the stopping criterion. If that node is not a head of family,

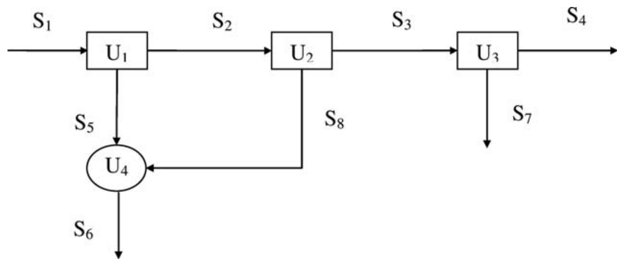


Figure 7. The mineral flotation process.

Table 2. Nominal Operation Condition for Mineral Flotation Process

Streams	1	2	3	4	5	6	7	8
F_i (kmol/hr)	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

continue exploring the next families. If that node is a head of family, disregarding all the nodes that are in the right-hand side and share the same root with that node. For example, if that node (which is a head of a family) is "123478910", then all the nodes that are on the right-hand side and share the same root ("1234") with that node shall be disregarded; the next node to explore is "12356789" (assuming there is ascending order in cost from Sensor 1 to Sensor 10).

(6) If *all* nodes in the current levels are either explored or disregarded, continue exploring the upper levels.

(7) The tree search terminates once it identifies a level where all nodes *are* found to be infeasible.

Hybrid vertical and level by level search

In this method, we combine breadth-first and depth-first strategies. The method is outlined next:

(1) Run the depth-first tree search method. Record the current best solutions and the associated depth level (number of sensors) in those solutions.

(2) Stop tree search if the number of nodes explored reaches the predefined limit. This limit depends on the size of the problem.

(3) The current best solution found is denoted as XQ and its depth level (the number of sensors) is denoted as Nle. The number of sensors in optimal solution is at most equal to Nle. Testing results (for medium problems) show that the number of sensors (depth level) in the optimal solution is generally in the range $[Nle - 2, Nle - 5]$.

(4) Switch to level traversal search.

(5) Choose the depth level to perform horizontal search to be one value in the range $[Nle - 1, Nle - 5]$, denoted as N.

(6) Explore horizontally all the nodes on the right-hand side of the branch that contains XQ and have the same depth level of N. These nodes are called root nodes.

(7) In each root node, check for its feasibility, then:

• If the current root node (level N) is feasible, then the nodes in the upper levels ($N - 1$, $N - 2$, etc.) can also be feasible. Then

- Explore the upper levels ($N - 1$, $N - 2$, etc.) by removing sensors out of the root node (we are exploring the parents only) with the following stopping criterion: stop exploring when the node is found to be infeasible.
- Do not explore the sister nodes in the same family with the current root node because even if these sister nodes are feasible, they result in the same nodes in the upper levels ($N - 1$, $N - 2$, etc.) as with the current root node.
- If that feasible node is head of a family, do not explore the families of nodes that share the same root and on the right-hand side of that head of family.
- If the current root node (level N) is infeasible.
- If its cost is lower than current best cost, explore the lower levels ($N + 1$, $N + 2$, etc.) but do not explore level Nle.
- If the cost is larger than current best cost, skip this node and the associated sister nodes of this current root node. If that node (which has higher cost than the current best) is head of a family, do not explore the families of nodes that share the same root and on the right-hand side of that head of family.

The procedure is depicted in Figure 6.

Examples

The proposed methods are implemented in Fortran running on a 2.8-GHz Intel Pentium 1028-MB RAM PC.

Example 1

Mineral flotation process. Consider a middle-scale example, the mineral flotation process (MFP) introduced by Smith and Ichiyen.[21] It is shown in Figure 7.

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, 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 flow rates and 16 compositions).

The nominal operating condition is given in Table 2 (taken from Narasimhan and Jordache[22]); the sensor costs are shown in Table 3.

Three design case studies described in Nguyen and Bagajewicz[14] are considered. The two level traversal tree search methods (the level by level search and hybrid search) are used to solve the problem. They both identify the optimal solution. The design specifications and the optimal solution are given in Table 4.

The performance of the two level traversal tree search methods are shown in Table 5 and compared with the performance of the depth-first tree search. The predefined limit to stop the depth-first tree search and switch to level traversal search is 500 (for level by level search) and 5000 (for hybrid search). In the hybrid search, the level to be explored horizontally is two levels above the level of the current best identified by the depth-first tree search (that is, $N = Nle - 2$).

It can be seen that the level by level search is generally better than the hybrid search. In the two design cases MFP2 and MFP3, the two level traversal search methods explored

Table 3. 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

Table 4. 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, and C_{7B}	F_1, C_{1A}, F_7 and 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}, and 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}, and C_{7B}
Sensors cost	1448	2118	2968

similar number of nodes, but the computational time of the level by level search is shorter than the other. The difference is largely due to implementation issue in Fortran of the hybrid search: from a node in the chosen level ($N = Nle - 2$), the tree search either goes up (explore upper levels $N - 1$, $N - 2$) or goes down (explore next levels $N + 1$, $N + 2$) by calling the appropriate subroutines. It is well known that before the commands in a subroutine are executed, a certain amount of time is spent to perform the preprocessing step (known in computer science as ''overhead''). The ''extra'' time spent on ''overhead'' explains why the computational time of hybrid method is longer than that of the level by level search. Although for design cases MFP1 and MFP2, the level by level search is not better than the equations-based method with decomposition, but if design case MFP3 is included for comparison, the level by level search can be considered to be better than the equations-based method, because the level by level search solved the design case MFP3 much faster. Another advantage of the level by level search over the equations-based method is that it is much simpler to use because it does not require any knowledge to decompose the problem (a poor choice of how to decompose the problem in equations-based method can lead to much longer computational time).

We now illustrate the detail of steps in the level by level search for the design Case 3 (MFP3). The list of sensors in ascending order of cost is [5, 3, 6, 1, 7, 2, 8, 4, 13, 17, 15, 20, 24, 18, 10, 22, 9, 11, 21, 14, 23, 16, 12, 19] (vector ***SC***). For simplicity, we use the indexes (or locations) of sensors in the vector ***SC*** to indicate the measurement locations. For example, if the active element in vector ***q*** (in Eqs. 1 and 2) is [1, 2, 3] then the actual chosen sensors (measurement location) are 5, 3, and 6 whose indexes in *SC* are 1, 2, and 3, respectively. The solution ***q*** = [123] has the smallest cost among all the solutions that have three sensors.

Let the set *R* be defined as follows: *R* = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. The depth-first tree search after exploring 500 nodes identifies [*R*, 14, 16, 17, 18, 19, 20, 22] as current best (containing 20 sensors and the current best cost is 3878, node *XQ*) at node 406. The node at which the depth-first tree search terminates (node *BQ*) is [*R*, 14, 16, *18*, *21*, *23*, *24*] (19 sensors). The first node to be explored by level by level search is the node that has the same depth level with the current best (20) and on the right-hand side of *XQ* and *BQ*. That node is [*R*, 14, 16, *19*, *20*, *21*, *22*, *23*] (the different part between this node and *XQ* is italicized). The node is also a head of family whose root is [*R*, 14, 16] and it has higher cost (cost is 3953) than current best. Thus, all nodes on the right-hand side and share the same root with this node are disregarded. The next node to be explored is [*R*, 14, *17*, *18*, *19*, *20*, *21*, *22*]. The same thing is observed (head of family with higher cost than current best), thus the next node to be explored is [*R*, *15*, *16*, *17*, *18*, *19*, *20*, *21*] (which is again a head of family; the roots of those heads of families are indicated by normal letters, the other members are indicated by italicized letters). The same thing is observed and this node is disregarded. There is no node in the current level (20) can compete with the current best. After exploring roughly 9000 nodes, the tree search completes exploring the two levels 20 and 19 (found a new current best at level 19, the new current best cost is 3653) and quickly moves to the next level (18). The first node to be explored in this current level (number of sensors = 18) is [*R*, 14, 16, 18, 22, *23*], this node has lower cost than the current best but it is infeasible, so its sisters node ([*R*, 14, 16, 18, 22, *24*]) is explored, which does not satisfy the stopping criterion either. The tree search keeps searching horizontally from left to right; in the process it visited nodes that have lower cost than current best but they are infeasible. The first node that is better than the current best is [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 16, 17, 18, *19*, *20*, *22*]. This new current best is not a head of family, so the next families are explored. The tree search continues in that fashion.

Table 5. Performance of Level Traversal Tree Search Methods, Mineral Flotation Process Example

Case Study		MFP1 (Low Spec.)	MFP2 (Moderate Spec.)	MFP3 (High Spec.)
Hybrid search	Computation time	4 min, 24 s	7 min, 22 s	11 min, 5 s
	Number of nodes explored	205,168	119,188	186,521
Level by level search	Computation time	1 min, 11 s	2 min, 52 s	6 min, 42 s
	Number of nodes explored	57,143	117,382	171,975
Depth-first tree search	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
Equations-based with decomposition	Computation time	13 s	40 s	1 h, 29 min
	Number of nodes explored	5077	13,622	200,245

We have shown how the searching process proceeds and how to eliminate nonoptimal nodes in the level by level search. The hybrid search is performed in the similar fashion. The only difference is that the hybrid search explores only one level (which is a chosen parameter). In each node in the chosen level, it either explores the parent (the root) of that node or the children originated from that node by calling the appropriate subroutines to either "go up" or "go down" the tree.

Discussion

It can be seen that the level traversal tree search is much more efficient than the depth-first tree search because it explores only the "promising" region. However, the fact that the current bests are found in the left side of the tree plays a significant part in reducing computational time because the tight bounds help to eliminate lots of nonoptimal solutions. If tight bounds are not obtained (in this context, when current bests are found in the right side of the tree), the level traversal tree search basically has to explore the whole level of tree, which makes it impossible to solve large-scale problems efficiently using this method. The large-scale problem with medium level of specification shown in Nguyen and Bagajewicz,[14] the TE example case study 2, exposes such limitation of the level traversal tree search. We attempted to solve the TE example using level traversal tree search, but the solutions provided by this method are worse than the solution obtained by the equation-based method with decomposition described in Nguyen and Bagajewicz[14] even after several days running. For the TE example, a kind of heuristic local search or approximate method is probably the most efficient method. This is part of ongoing work.

One of the main reasons why this methodology is computationally efficient is because there is monotonicity in objective values (sensor costs) of nodes belonging to a same family of nodes (cost increases from left to right), this property is exploited to eliminate nonoptimal solutions in the tree search procedure. The methodology cannot be applied if observability, redundancy, or precision is utilized as objective function (e.g., maximizing precision), because in such case, there is no monotonicity in objective values of nodes belonging to a same family of nodes.

Conclusions

In this article, we have explored the efficacy of a new level traversal search in the tree of solutions for the instrumentation network design/upgrade problem. The method helps to reduce computation time of the depth-first tree search by skipping the nonfeasible region and intelligently disregarding the nonoptimal solutions via notion of families of nodes. The method is very efficient if feasible solutions are found in the left-hand side of the tree.

Literature Cited

1. Bagajewicz M. *Process Plant Instrumentation. Design and Upgrade*. Boca Raton, FL: CRC Press, 2000.
2. Kretsovalis A, Mah RSH. Effect of redundancy on estimation accuracy in process data reconciliation. *Chem Eng Sci.* 1987;42:2115–2121.
3. 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:S535–S540.
4. Madron F, Veverka V. Optimal selection of measuring points in complex plants by linear models. *AIChE J.* 1992;38:227–236.
5. Bagajewicz M. Design and retrofit of sensors networks in process plants. *AIChE J.* 1997;3:2300–2306.
6. Bagajewicz M, Cabrera EA. New MILP formulation for instrumentation network design and upgrade. *AIChE J.* 2001;48:2271–2282.
7. Chmielewski D, Palmer T, Manousiouthakis V. On the theory of optimal sensor placement. *AIChE J.* 2002;48:1001–1012.
8. Kelly JD, Zyngier D. A new and improved milp formulation to optimize observability, redundancy and precision for sensor network problems. *AIChE J.* 2008;54:1282–1291.
9. Sen S, Narasimhan S, Deb K. Sensor network design of linear processes using genetic algorithms. *Comput Chem Eng.* 1998;22:385–390.
10. 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.
11. Carnero M, Hernandez J, Sanchez M, Bandoni A. On the solution of the instrumentation selection problem. *Ind Eng Chem Res.* 2005;44:358–367.
12. Gala M, Bagajewicz MJ. Rigorous methodology for the design and upgrade of sensor networks using cutsets. *Ind Eng Chem Res.* 2006;45:6687–6697.
13. Gala M, Bagajewicz MJ. Efficient procedure for the design and upgrade of sensor networks using cutsets and rigorous decomposition. *Ind Eng Chem Res.* 2006;45:6679–6686.
14. Nguyen DQ, Bagajewicz M. Design of nonlinear sensor networks for process plants. *Ind Eng Chem Res.* 2008;47:5529–5542.
15. Bagajewicz M, Sánchez M. Design and upgrade of non-redundant and redundant linear sensor networks. *AIChE J.* 1999;45:1927–1939.
16. Bagajewicz M, Sánchez M. Cost-optimal design of reliable sensor networks. *Comput Chem Eng.* 2000;23:1757–1762.
17. Bagajewicz M, Sánchez M. Reallocation and upgrade of instrumentation in process plants. *Comput Chem Eng.* 2000;24:1961–1980.
18. Bagajewicz M, Sánchez M. On the impact of corrective maintenance in the design of sensor networks. *Ind Eng Chem Res.* 2000;39:977–981.
19. Bagajewicz M, Sánchez M. Duality of sensor network design models for parameter estimation. *AIChE J.* 1999;45:661–664.
20. Diwekar U. *Introduction to Applied Optimization (Springer Optimization and Its Applications), 2nd ed.* NY, USA: Springer, 2008.
21. Smith HW, Ichiyen N. Computer adjustment of metallurgical balances. *Can Znst Mining Metall (C.Z.M.) Bull.* 1973;66:97–100.
22. Narasimhan S, Jordache C. *Data Reconciliation and Gross Error Detection: An Intelligent Use of Process Data*. Houston, Texas: Gulf Publishing, 2000.

Manuscript received Oct. 24, 2009, and revision received May 2, 2010.