PERGAMON Mathematical and Computer Modelling 0 (2005) 1–0
www.elsevier.com/locate/mcm

Discrete Sensor Placement Problems in Distribution Networks

T. Y. Berger-Wolf
Center for Discrete Mathematics and Theoretical Computer Science (DIMACS)
CoRE Bldg., Rutgers University
96 Frelinghuysen Rd., Piscataway, NJ 08854, U.S.A.
tanyabw@dimacs.rutgers.edu

W. E. Hart
Computer Science Research Institute
Sandia National Laboratories
Albuquerque, NM 87185, U.S.A.
wehart@sandia.gov

J. Saia
Department of Computer Science
University of New Mexico
Albuquerque, NM 87131, U.S.A.
saia@cs.unm.edu

(Received December 2003; revised and accepted March 2005)

Abstract—We consider the problem of placing sensors in a network to detect and identify the source of any contamination. We consider two variants of this problem:

(1) *sensor-constrained*: we are allowed a fixed number of sensors and want to minimize contamination detection time; and
(2) *time-constrained]*: we must detect contamination within a given time limit and want to minimize the number of sensors required.

Our main results are as follows. First, we give a necessary and sufficient condition for source identification. Second, we show that the sensor and time constrained versions of the problem are polynomially equivalent. Finally, we show that the sensor-constrained version of the problem is polynomially equivalent to the asymmetric k-center problem and that the time-constrained version of the problem is polynomially equivalent to the dominating set problem.

1. INTRODUCTION

We consider the problem of placing sensors in a building or a utility network in order to detect contamination of the air or water supply. Practical motivation for this problem derives from recent

We are grateful to C. Phillips and the anonymous reviewers for their helpful comments. This work was performed in part at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000. This work is supported by the National Science Foundation postdoctoral fellowship Grants EIA 02-03584, EIA 02-05116 (T. Berger-Wolf), by the National Science Foundation Grant CCR-0313160 (J. Saia), and by the Sandia University Research Program Grant No. 191445 (J. Saia).

0895-7177/05/$ - see front matter Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX
PII:00

world events including Tokyo's subway incident, London's poison gas bomb plot, and various U.S. government warnings. While more effective sensors (e.g., the SnifferSTAR air quality sensor [1]) are currently being developed to address increasing threats of contamination, these new sensors are likely to be expensive. Thus, algorithmic techniques are needed to place sensors in a network in such a way that cost is minimized and contamination can still be quickly detected.

Two possible goals for sensor placement are:

(1) *contamination detection*, i.e., ensuring quick detection of a contamination event, and
(2) *source identification*, i.e., ensuring that the source of contamination can always be identified.

Two natural constraints for sensor placement are:

(1) *sensor-constrained*, i.e., allowing only a fixed number of sensors and
(2) *time-constrained*, i.e., requiring contamination detection or source identification within a given time limit.

These two goals and two constraints define four sensor placement problems which we address in this paper.

Our results, in this paper, provide the first analysis of the computational complexity of these sensor placement problems. The problems are described formally in Section 2.1. For contamination detection, we show that the sensor-constrained and time-constrained problems are polynomially equivalent and NP-hard (Section 2.2). We show that the sensor-constrained contamination detection problem is polynomially equivalent to the asymmetric k-center problem (Section 4) and the time-constrained contamination detection problem is polynomially equivalent to the dominating set problem (Section 5). In addition, we describe necessary and sufficient conditions for source identification (Section 3); and give exact solutions to two specific cases of the source identification problem: the uniform clique and rooted trees (Section 6).

A variety of numerical techniques have been proposed for sensor placement. There are several integer programming formulations for contamination detection in water networks [2–5]. Berry *et al.* [2] show that these integer programs can be robust to noise in data. Unfortunately, integer programming can be exponential slow so these formulations of the problem may have difficulty scaling to real-world problems. Numerical methods have also been developed for the problem of source identification [6–10]. The approaches suggested in these papers are either to place sensors randomly or apply a greedy heuristic to compute a minimal (but not necessarily optimal) set of sensors capable of contamination source identification. Unfortunately, these numerical methods can also be exponentially slow.

There have also been nonnumerical approaches to sensor placement. González-Banos and Latombe show how to place visual sensors in a building for 3D mapping using a combinatorial optimization approach [11]. Additionally, heuristic optimizers have been used to perform sensor placement in water networks using water quality and protection measures based on detailed hydraulic models like EPANET (e.g., see [12–15]).

2. DISCRETE OPTIMIZATION MODEL

2.1. Overview

In this section, we give a formal description of our sensor placement problems. We model the network as a directed weighted graph $G = (V, E)$. V is a set of vertices representing possible locations for the sensors. These may be rooms in a building or pipe junctions in a water network. E is a set of edges representing flow between the vertices. These may be airways or hallways in a building or pipes in a utility network. Each edge (i, j) has weight $r_{ij} \in \Re^+$ which is the time it takes for contaminant to pass from vertex i to vertex j. We use the shortest path metric to define the actual translocation rates in the graph. That is, for any two vertices i and j, the

translocation rate r_{ij} is defined as the length of the shortest path between i and j; r_{ij} is infinite if no such path exists. Since the shortest path metric defines the actual translocation rates in this graph, the triangle inequality is valid. We can construct an adjacency matrix representation of the effective translocation rates in the graph by using any all-pairs shortest paths algorithm for directed weighted graphs with no negative weights[1]. Thus, the input graph is a weighted complete graph (possibly with some infinite weight edges), where some edges represent the actual flow conduits and the rest are inferred to represent the effective translocation rates. For simplicity of presentation, however, in some cases we will explicitly describe only the edges representing flow conduits and the remaining edges are inferred.

Given this input, the goal is to place sensors on the vertices so it is always possible to detect the contamination and, additionally, identify the vertex that is the source of contamination. We say that a sensor s can detect contamination at vertex v if there exists a finite length directed path from v to s. We say that s detects contamination at v within time t if the length of the v-s path is at most t.

We make several assumptions about the sensor placement problem. First, we assume that a sensor can detect the presence of contaminant before it reaches dangerous levels and that once a sensor is activated, it remains activated. Second, we assume rapid mixing in the air flow. That is, (1) there is no delay between the material entering and leaving a compartment, and (2) the spread of the material inside the compartment is even. Third, we assume that there is a single contamination source. Otherwise, a sensor is needed at every vertex for source identification to be possible. For any particular contamination event, we assume that all the vertices down the flow from the contamination source eventually become contaminated. Any vertex in the network may be a potential contamination source.

In the sensor-constrained variant of our problem, we are given a maximum number of sensors, $S_{\max}$, and we want to minimize the time from contamination to detection or source identification. In the time-constrained variant of the problem, we are given a time limit T, and want to minimize the number of sensors required to detect contamination or identify the source within that time limit.

2.2. Sensor-Constrained and Time-Constrained Equivalence

We now show a polynomial time equivalence between the sensor-constrained and time-constrained variants of the sensor placement problem.

Observation 1. *The sensor-constrained and time-constrained versions of the sensor placement problem are polynomially equivalent. Moreover, an algorithm for the sensor-constrained version can be converted to an algorithm for the time-constrained version with a* $\lg |V|$ *factor increase in the running time, and an algorithm for the time-constrained version can be converted to an algorithm for the sensor-constrained version with a* $\lg m$ *factor increase in the running time (where* $m \leq n^2$ *is the number of finite weight edges).*

Proof. Suppose we have an exact algorithm that solves the sensor-constrained version of the problem. We now describe an algorithm for the time-constrained version, with time limit T, that minimizes the number of sensors. Set the sensor limit at $|V|$ and use the given algorithm to find the minimum time to detection or identification. If the solution for the contamination source identification exceeds T, then there is no feasible solution to the identification problem. (Note that when sensor limit is $|V|$ the time to detection is 0, and therefore, there is a feasible solution.) Otherwise, use binary search on the sensor limit to find the smallest number of sensors for which the solution to the sensor-constrained version is within the specified time limit T. There are at most $\lg |V|$ iterations of the binary search, hence, the logarithmic factor increase in the running time.

[1]The best algorithm to date is, we believe, Pettie's algorithm that runs in $O(|E||V| + |V|^2 \log\log |V|)$ time [16].

The other direction is very similar. Suppose we have an exact algorithm for the time-constrained version. Sort the edges by their length. (Note that this adds an $O(m \lg m)$ *additive* factor to the running time, which is assumed to be less than $A_{t-c} \times \lg m$, where A_{t-c} is the running time of the algorithm for the time-constrained problem.) Recall that we assume that all the edges exist and their weights are defined by the shortest path metric. Therefore, the time to detection or identification for any algorithm must be the length of some edge. Start with the time limit set at $r_{\max}$, the length of the longest finite edge. Solve this problem for the smallest number of sensors. This gives the smallest feasible number of sensors. Thus, if the solution requires more sensors than the sensor limit $S_{\max}$, then there is no feasible solution. Otherwise, use binary search on edge lengths to find the smallest time limit for which the time-constrained solution gives at most the number of alloted sensors.

Note that the binary search approach assumes monotonicity. However, it is clear that the number of sensors needed monotonically increases as the detection time limit decreases and vice versa. ∎

3. SOURCE IDENTIFICATION

In both the sensor and time-constrained variants of sensor placement, the question of unique contamination source identification must be addressed. The problem of source identification is clearly at least as hard as contamination detection, since detection is necessary for identification. The following lemma gives a necessary and sufficient condition for source identification.

Lemma 1. *Unique contamination source identification is possible within the system if and only if every vertex has a unique configuration of sensors and reaction time delays between successive sensors. That is, let $S = \{s_0, \ldots, s_{k-1}\}$ be the set of all sensors in the graph. For a vertex v_i, let $(s_{i_0}, \ldots, s_{i_{k-1}})$ be the sensors ordered by the length of the path from vertex v_i to each sensor (breaking the ties by sensor number). Further, let $(d_{i_1}, \ldots, d_{i_{k-1}})$ be the reaction delay time sequence, where d_{i_j} is the difference between the path length from v_i to $s_{i_{j-1}}$ and the path length from v_i to s_{i_j}. If there is no path from v_i to s_{i_j}, then $d_{i_j} = \infty$. Then, each vertex must have a unique tuple $((s_{i_0}, 0), (s_{i_1}, d_{i_1}), \ldots, (s_{i_{k-1}}, d_{i_{k-1}}))$.*

The reaction time delay sequence in addition to the sensor permutation is generally necessary for unique identification. Figure 1 shows an example of the same sensor permutation with different time delays that allows unique identification. (The edges not drawn in the figure are inferred.)

Clearly, there must be a path from each vertex to at least one sensor for source identification. In the worst case, the entire sequence of sensors may be necessary for the unique source identification. Figure 2 shows an example of this phenomena, with six vertices and three sensors that can be easily generalized to s sensors with $2s$ vertices for an arbitrary s.

Given a graph with edge weights and a sensor placement, using Lemma 1, it is possible to verify whether the placement indeed allows unique source identification. It also leads to a natural contamination source identification algorithm. First, we construct the sensor-delay sequence for each vertex by sorting the set of weights from a vertex to all the sensors. The sorting and computing of the delay in the reaction time between two successive sensors can be done in $O(|V||S| \log |S|)$ time, where $|S|$ is the number of sensors. Once the sensor-delay sequences are constructed, we need to verify that each vertex has at least one finite sensor reaction time and that there are no two identical sequences. This can be done in a straight-forward way in $O(|V|^2|S|)$ time (in the worst case, the sequences differ in the last delay time). Now we can build a delay sequence multigraph, with sensors as vertices and edges (i, j) of weight w_{ij} if there exists a sensor reaction sequence in which sensor j reacts after sensor i with a delay of w_{ij}. For each edge we also keep a label that identifies the set of possible vertices that are consistent with the delay sequence as being the contamination source. Traversing two consecutive edges implies taking the intersection of their labels. This graph can be built in $O(|V||S|)$ time. Thus, the total

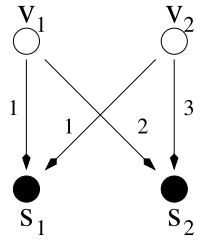

Figure 1. Sensors s_1 and s_2 will react in the same order to vertices v_1 or v_2 being contaminated. However, if v_1 is the contamination source, then the reaction delay between s_1 and s_2 is 1, while if v_2 is the source, the delay is 2.

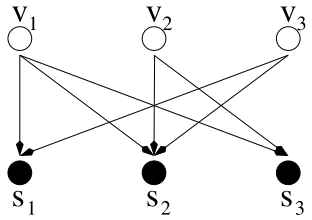

Figure 2. An example of a graph where an entire sensor sequence may be necessary for a unique contamination source identification. For each sensor s_i the incoming edges have weight i. The delay sequence for the vertices is as follows: $v_1 : (s_1, 0), (s_2, 1), (s_3, 1)$, $v_2 : (s_2, 0), (s_3, 1)$, $v_3 : (s_1, 0), (s_2, 1)$. For each sensor vertex s_i the delay sequence is $(s_i, 0)$. The minimum number of sensors for this graph is the three sensors shown in the figure.

preprocessing time is $O(|V|^2|S|)$. When contamination occurs, then any actual sensor reaction sequence corresponds to a unique path in the delay sequence graph. Thus, the contamination source can be identified in $O(|S|)$ time. We have shown the following statement.

PROPOSITION 1. *Given a placement of $|S|$ sensors in the system, it takes at most $O(|V|^2|S|)$ time to verify that unique source contamination identification is possible and, in case of contamination, it takes at most $O(|S|)$ time to identify the source.*

Note that this algorithm relies on the relative difference of the edge weights, and not on their particular values. If the error in measurement of the translocation rates is less than the smallest difference between any two paths, the algorithm robustly identifies sources. To our knowledge, this sort of robustness has not been quantified in other methods for source identification for these problems.

4. SENSOR-CONSTRAINED CONTAMINATION DETECTION

In the sensor-constrained contamination detection problem, we are given a weighted directed graph $G = (V, E)$ with positive weights r_{ij} and a positive integer $S_{\max}$, which is the maximum number of sensors to be used. Our goal is to place the sensors onto the vertices of the graph in a way that minimizes the maximum contamination detection time. This problem is equivalent to the asymmetric k-center problem, which is well known to be NP-hard [17].

ASYMMETRIC k-CENTER. We are given a complete directed graph $G = (V, E)$ of shortest (weighted) path distances between the vertices that satisfies triangle inequality, and a positive integer k. We must find a subset of vertices S, $|S| = k$, which minimizes the longest distance from a vertex in S and any vertex in the graph. That is we want to find appropriate S minimizing $\text{cost}(S)$, defined as follows:

$$\text{cost}(S) = \max_{v \in V} \min_{s \in S} \text{dist}(s, v).$$

The asymmetric k-center problem cannot be approximated within a factor of $(1 - o(1)) \log^* n$ unless NP $\subseteq$ DTIME($n^{\log \log n}$). It is also inapproximable within any constant factor unless

P=NP [18]. There exist both an $O(\log^* n)$-approximation algorithm [19] and a $O(\log^* k)$-approximation algorithm [20], which are theoretically the best possible (unless NP $\subseteq$ DTIME($n^{\log\log n}$)).

THEOREM 1. *The sensor-constrained contamination detection problem is equivalent to the asymmetric k-center problem.*

PROOF. To show the reduction in either direction, we equate $k = S_{\max}$ and retain the underlying directed graph G with the edge weights reversed $\text{dist}(i,j) = r_{ji}$. Since the edge weights use the shortest path metric, the triangle inequality is satisfied. A solution to the asymmetric k-center problem is a subset of at most k vertices, S, that minimizes the distance from a vertex in S to a vertex in the graph. That is, this is a subset of at most $S_{\max}$ vertices such that the maximum time from any vertex to a vertex in S is minimizes among all such subsets. That is, this is a subset of vertices such that a placement of sensors at these vertices minimizes the contamination detection time. ∎

5. TIME-CONSTRAINED CONTAMINATION DETECTION

In the time-constrained contamination detection problem, we are given a weighted directed graph $G = (V, E)$ with positive weights r_{ij} and a positive integer T, which is the detection time limit. Our goal is to place the minimum number of sensors onto the vertices of the graph that allows contamination detection and source identification within the time limit. From Observation 1, we know that the time-constrained and sensor-constrained variants are polynomially equivalent. We have shown in Section 4 that the sensor-constrained variant is NP-hard, therefore, the time-constrained variant is NP-hard as well. The question remains, however, what is the best approximation algorithm for the time-constrained variant.

Notice, that we cannot directly apply any approximation algorithm for the sensor-constrained minimization using the binary search to find the smallest number of sensors within a given time limit. The solution that we find might exceed the specified limit. Thus, the approximate solution must be within the time limit. However, in that case, we might restrict the problem too much, increasing the number of sensors needed. In fact, for an arbitrary graph, there is no bound on the increase in the number of sensors with the decrease in the time detection limit. Consider a uniform clique with all the edge weights (in both directions) being T, the time limit. We need only one sensor to guarantee contamination detection within time T. However, detection within any time less than T requires all n sensors. The clique graph is not unique. We can construct numerous nontrivial examples with arbitrary increase in the number of sensors when the detection time limit decreases. Thus, we need an approximation algorithm designed specifically for the time-constrained version of the problem. For that, we reduce time-constrained sensor placement directly to the minimum dominating set problem.

MINIMUM DOMINATING SET. Given a graph $G = (V, E)$ find the smallest subset $S \subseteq V$ such that for all $u \in V - S$, there is a $v \in S$ such that $(u, v) \in E$.

Minimum dominating set is not approximable within $c \ln |V|$ for any constant $0 < c \leq 1$ [21]. The best approximation algorithm is the 1974 Johnson's $1 + \ln |V|$ approximation algorithm [22] which is the direct application of the greedy set cover algorithm. The running time of the greedy set cover algorithm is asymptotically proportional to the sum of the set sizes. In case of the minimum dominating set problem, there are $|V|$ sets, each of size at most Δ (maximum degree of a vertex in the graph). Thus, the running time is $O(|V|\Delta)$.

THEOREM 2. *Time-constrained sensor placement problem is polynomially equivalent to minimum dominating set.*

PROOF. Given a weighted graph $G = (V, E)$ for the sensor placement problem and a time limit T, we create the input graph $G' = (V, E')$ for the minimum dominating set problem as follows. An

edge $(u, v) \in E'$ if and only if $r_{u,v} \leq T$. That is, we create the "reachability in T graph". A solution to the minimum dominating set on G' provides a minimum subset of vertices $S \subseteq V$ such that for any $u \in V$, it is either in S or there exists $v \in S$ such that $(u, v) \in E'$. That is, S is a minimum subset of vertices V such that for any vertex in the original graph G the distance from it to S is at most T. This is the solution to the time-constrained sensor placement problem for contamination detection.

For the other direction, given graph $G = (V, E)$ for the minimum dominating set problem, we create an instance of the time-constrained sensor placement problem by assigning all the edges in E weights 1 and setting the time limit $T = 1$. The weights for the edges not present in E are the path length between the corresponding vertices. A solution to the time-constrained sensor placement problem provides a minimum subset of vertices S such that for any vertex $u \in V - S$ the distance from u to a vertex in $v \in S$ is at most 1. That is, $(u, v) \in E$. ∎

6. SOURCE IDENTIFICATION IN SPECIAL GRAPHS

In this section, we give exact solutions to the source identification problem for two special graphs: the uniform clique and rooted trees.

6.1. Uniform Clique

Consider the uniform clique graph on n vertices, K_n, where each pair of vertices i, j is connected by a bidirectional edge with weight r. The complete solution to the uniform clique graph sensor placement problem is outlined in the following two observations.

OBSERVATION 2. *The uniform clique graph K_n needs one sensor for contamination detection (in time greater than 0) and a sensor placed at any vertex guarantees detection within the time limit of r.*

OBSERVATION 3. *The uniform clique graph K_n needs at least $n - 1$ sensors for unique source identification (in time greater than 0) and any $n-1$ sensors guarantee unique source identification within the time limit of r.*

To reduce the detection or identification time to 0, it is necessary to place a sensor at every vertex. This is true for any graph with all nonzero edge weights. Clearly, any sensor placement graph is a weighted clique (with possibly some infinite edge weights).

6.2. Rooted Trees

A *rooted tree* is a tree graph with a special vertex designated as a root. All the edges are oriented away from the root. That is for any edge (i, j), it is oriented from i to j if i is on the path from the root to j. Water and other distribution networks can in some cases be modeled as rooted trees. In such a network there is a single supply source and it is delivered along a unique path to each destination. This model does not take into account possible back flow, that is, it assumes the flow moves from the source to the destinations only. (The rest of the edges in the graph are inferred from the tree edges.)

6.2.1. Detection

There must be a sensor in each leaf of the tree (a vertex with no outgoing edge), otherwise there is no way to detect contamination at that vertex. For a detection time limit T, we use algorithm *rtree* presented in Figure 3 to find the minimum set of sensors that ensures contamination detection within time T.

RTREE	
(1)	Place a sensor at each leaf.
(2)	Follow the tree edges towards the root from the current sensors and mark all the vertices that have distance at most T to a sensor as "covered".
(3)	Put a sensor at each uncovered vertex that is first on the path from a sensor to the root.
(4)	Repeat Steps 2 and 3 until all vertices are covered.

Figure 3. Algorithm rtree for contamination detection in rooted trees.

THEOREM 3. *Algorithm rtree produces a minimum set of sensors that guarantees contamination detection within the specified time limit T.*

PROOF. Suppose to the contrary that an optimal set of sensors is smaller than that produced by the algorithm. There must exist a sensor in that set that was placed at a vertex already covered at some iteration of the algorithm. Let s be the first such sensor. The length of the path from s to the closest sensor other than s is less than the time limit T. Notice that in the set of sensors produced by rtree each two sensors are at least distance T apart, since we never place a sensor on a covered vertex. In a tree graph, there is a unique path from the root to every vertex. If we replace the sensor s with the next sensor s_R from rtree on the path from the root to s, then we do not increase the number of sensors. Moreover, all the vertices are still covered since s was the vertex furthest from the root that was not in the set produced by rtree. When we replace s by s_R, all the descendants of s_R are still covered by the set of sensors that is the same in the optimal and rtree. The ancestors of s are covered because s_R is the first vertex uncovered by that set and it is closer to the next optimal sensor on the path from the root to s. Thus, we can replace the optimal set of sensors with that produced by the algorithm without increasing the number of sensors. This is a contradiction to the assumption that the optimal set is smaller than the one produced by the algorithm. Hence, the algorithm rtree produces an optimal set of sensors. ∎

6.2.2. Identification

Notice that the above algorithm guarantees *detection* within the specified time limit, it does not guarantee identification. We need to modify the algorithm slightly to guarantee unique contamination source identification.

LEMMA 2. *In a rooted tree graph, for any vertex at most two sensors are sufficient to uniquely identify contamination at that vertex (the two sensors are not necessarily the same for all the vertices).*

PROOF. In a tree with no vertices of degree 2, every internal vertex has a set of descendents different from the set of descendents of its child (besides the child itself), and each internal vertex has at least one pair of descendants such that it is their least common ancestor[2]. Thus, for any internal vertex it is sufficient to have sensors at some two vertices whose least common ancestor it is. These two sensors identify their least common ancestor as the contamination source in such a tree. Placing a sensor at any vertex of degree 2 uniquely identifies contamination at that vertex and effectively converts the tree to a tree with no vertices of degree 2. ∎

Thus, when no time limit is imposed, it is sufficient to place the sensors at the leaves to ensure unique contamination source identification in a tree with no degree-2 vertices. However, we must

[2] *The least common ancestor* of two nodes in a rooted tree is the node that is an ancestor of both and is the furthest of all such from the root.

place a sensor at any vertex that has only one outgoing edge, since otherwise it cannot be a least common ancestor of some two vertices. With these observations in mind, in Figure 4, we present the modified algorithm *rtree-id* that produces the smallest set of sensors that guarantees unique source identification within a given time limit T.

RTREE-ID	
(1)	Place a sensor at each vertex with out-degree less than 2.
(2)	Follow the edges in the reverse direction from the current sensors and mark all the vertices that have distance at most T to a sensor. For each vertex that is marked at least twice, mark it as "covered".
(3)	Put a sensor at each not "covered" vertex that is first on the path from a sensor to the root.
(4)	Repeat Steps 2 and 3 until all vertices are covered.

Figure 4. Algorithm rtree-id for contamination source identification in rooted trees.

THEOREM 4. *Algorithm rtree-id produces a minimum set of sensors that guarantee contamination source identification within the specified time limit.*

The proof is similar to that of Theorem 3.

Both algorithms rtree and rtree-id minimize the number of sensors given a time limit, that is, they are algorithms for the time-constrained model of the problem. However, since these algorithms are exact, we have shown in Section 2.2 that using those algorithms, we can construct a solution to the sensor-constrained problem with a $O(\log |V|)$ factor increase in running time. Thus, we have solved the sensor placement problem for rooted trees.

6.3. Directed Acyclic Graphs

A *directed acyclic graph* (DAG) is a directed graph with no cycles. The vertices with no incoming edges are called *maximal* elements, and the vertices with no outgoing edges are called *minimal* elements of the graph. Many distribution networks can be realistically modeled as DAGs. In such a network there may be many supply sources (maximal elements), but the flow is only from the source to the destinations. This model still does not take into account possible back flows.

6.3.1. Detection

THEOREM 5. *Time-constrained sensor placement for contamination detection in a DAG is NP-complete.*

PROOF. The general sensor placement problem for contamination detection is in NP, hence, this special case is in NP as well. We show NP-hardness by a reduction from minimum set cover.

MINIMUM SET COVER. Given a universe $U = \{1, 2, \ldots, n\}$ and a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ such that $S_i \subseteq U$, find the smallest number of sets in $\mathcal{S}$ such that $\bigcup_j S_j = U$.

Given an instance of the minimum set cover problem with the elements $U = \{1, 2, \ldots, n\}$ and a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$, we create an instance of the time-constrained sensor placement on a DAG as follows. There are three types of vertices. All the maxima vertices correspond to U, one vertex for each element. There is also a vertex for each set, with an edge from an element u to a set S if $u \in S$. Finally, there is one minimum vertex with an edge from each of the sets to it. Clearly, the resulting graph has no directed cycles and can be constructed in polynomial time. Figure 5 illustrates the construction.

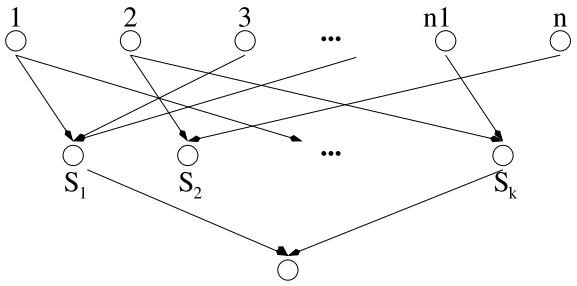

Figure 5. A graph obtained after the reduction from an instance of set cover. There is an edge from i to S_j if $i \in S_j$. All vertices S_j have an edge to the minimum vertex.

To finish the reduction, all the edges have the same weight 1 and the time limit for detection is $T = 1$.

Lemma 3. *There exists a feasible set cover of the elements of U if and only if there exists a feasible solution to resulting sensor placement problem with no sensors placed at the maximal vertices.*

Proof. Given a feasible set cover of U, we place the sensors at the minimal vertex and the vertices corresponding to the sets in the cover. Contamination in the minimum vertex and every vertex S_i is detected by the sensor at the minimum vertex (within time 1). Contamination at any maximal vertex i is detected by the sensor at the vertex corresponding to the set S_j from the set cover that contains i.

Conversely, given a placement of sensors in the graph with no sensors at the maximal vertices, we create a set cover by taking the sets corresponding to the vertices S_j that have sensors placed at them. Contamination at any of the maximal vertices must be detected by one of these sensors, since the distance to the minimum vertex is 2. Thus, the sets corresponding to these vertices cover all the elements of U. ∎

We now continue with the proof of Theorem 5. If there exists a feasible solution to the sensor placement problem with no sensors at the maximal vertices, then there exists an optimal solution with no sensor at the maximal vertices. This is due to the fact that we can replace any sensor at a maximal vertex i with a sensor at a vertex S_j that has an edge from i with no increase in the total cost.

Given an optimal solution to the sensor placement problem with no sensor at the maximal vertices, we construct a set cover by taking the sets corresponding to the vertices S_j with sensors. From Lemma 3, this is a feasible solution to the set cover problem. This is also an optimal solution, since the number of sensors in any sensor placement with no sensor at the maximal vertices is one more than the size of the set cover.

Thus, the sensor placement problem on a DAG is NP-complete. ∎

7. CONCLUSIONS AND DISCUSSION

We have defined discrete models for the problem of placing sensors in distribution networks and presented robust approximation methods. Our results show that optimal solutions cannot be efficiently found in all cases (unless P = NP), but that near-optimal solutions can be efficiently generated. Although we have not applied the approximation methods we discuss, it is clear that fast approximation methods are important for sensor placement because real-world data sets can be quite large. For example, a water network for a large municipality can easily contain over 10,000 major distribution pipes. We have also addressed the important goal of the identification of the source of contamination for the first time in a discrete setting.

The models considered in this paper make many simplifications and assumptions that might need to be revised for practical applications. For example, we have assumed a single set of

network flows. Although this might be reasonable for building ventilation systems, this will often be insufficient for large-scale municipal water networks. Network flows in water networks vary based on consumer demands, which naturally vary throughout the day and by day of the week. Consequently, sensor placement methods for such domains need to account for the best placement for a variety of flows (e.g., see the model proposed by Berry *et al.* [2]). Even if the gross flow remains stable, there will naturally be modest variations in flows. A more careful consideration of these effects needs to be addressed to ensure quick robust contamination detection and source identification.

Another major simplification in our model is that we have assumed robust detection regardless of the contaminant concentration level. A more realistic model would require a minimum concentration for robust detection. Additionally, it may be necessary to explicitly model sensor failures. Sensor failures might simply reflect the ability of sensors to operate under normal operating conditions. However, modeling this aspect could account for malicious destruction of sensors (e.g., by assessing how many sensors could be destroyed without seriously compromising their detection capacity).

Finally, we note that accurate prediction of sensor placements might require explicit models of the decay of contaminants within the network. For example, it is well known that chemicals like chlorine bind to water pipes, which is modeled by water models such as EPANET [23]. Contaminants that decay rapidly will naturally be more difficult to detect, though this is not captured in our models.

REFERENCES

1. N. Singer, Flying SnifferSTAR may aid civilians and US military, *Sandia LabNews* **55** (2).
2. J. Berry, L. Fleischer, W.E. Hart and C. Phillips, Sensor placement in municipal water networks, In *Proc. World Water and Environmental Resources Conference*, 2003.
3. J. Berry, W.E. Hart, C.A. Phillips and J. Uber, A general integer-programming-based framework for sensor placement in municipal water networks, In *Proc. Sixth Annual Symposium on Water Distribution Systems Analysis*, 2004.
4. R.D. Carr, H.J. Greenberg, W.E. Hart and C.A. Phillips, Addressing modelling uncertainties in sensor placement for community water systems, In *Proc. Sixth Annual Symposium on Water Distribution Systems Analysis*, 2004.
5. J.P. Watson, W.E. Hart and H.J. Greenberg, A multiple-objective analysis of sensor placement optimization in water networks, In *Proc. Sixth Annual Symposium on Water Distribution Systems Analysis*, 2004.
6. A. Birchall, A microcomputer algorithm for solving compartmental models involving radionuclide transformations, *Health Physics* **50** (3), 389–397, (1986).
7. A. Birchall and A.C. James, A microcomputer algorithm for solving first-order compartmental models involving recycling, *Health Physics* **56** (6), 857–868, (1989).
8. F. Gelbard, J.E. Brockmann, K.K. Murata and W.E. Hart, An algorithm for locating sensors in a large multiroom building, Tech. Rep. SAND2000-0851, Sandia National Laboratories, (2000).
9. C.D. Laird, L.T. Biegler, B.G. van Bloemen Waanders and R.A. Bartlett, Time dependent contamination source determination for municipal water networks using large scale optimization, *Journal of Water Resources Planning and Management* (to appear).
10. C.D. Laird, L.T. Biegler, B.G. van Bloemen Waanders and R.A. Bartlett, Time dependent contamination source determination: A network subdomain approach for very large networks, In *Proc. World Water and Environmental Resources Congress*, (2004).
11. H. González-Banos and J.-C. Latombe, A randomized art-gallery algorithm for sensor placement, In *Proceedings of the 17^{th} Annual Symposium on Computational Geometry (SoCG)*, pp. 232–240, ACM Press, New York, NY, (2002).
12. A. Kessler, A. Ostfeld and G. Sinai, Detecting accidental contaminations in municipal water networks, *J. Water Resources Planning and Management*, 192–198, (1998).
13. A. Kumar, M. Kansal and G. Arora, Discussion of "Detecting accidental contaminations in municipal water networks", *Journal of Water Resources Planning and Management*, 308–310, (1999).
14. A. Ostfeld and E. Salomons, Optimal layout of early warning detection stations for water distribution systems security, *Journal of Water Resources Planning and Management* **130** (5), 377–385, (2004).
15. M. Tryby, D. Boccelli, J. Uber and L.A. Rossman, Facility location model for booster disinfection of water supply networks, *Journal of Water Resources Planning and Management*, 322–332, (2002).

16. S. Pettie, A faster all-pairs shortest path algorithm for real-weighted sparse graphs, In *Automata, Languages and Programming. 29^{th} International Colloquium. Proceedings, Volume 2380 of Lecture Notes in Computer Science*, (Edited by P. Widmayer, F. Triguero, R. Morales, M. Hennessy, S. Eidenbenz and R. Conejo), pp. 85–97, ICALP 2002, Malaga, Spain, July 8–13, 2002, Springer, Berlin, (2002).
17. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, (1979).
18. J. Chuzhoy, S. Guha, E. Halperin, S. Khanna, G. Kortsarz and J. Naor, Asymmetric k-center is $\log^* n$ hard to approximate, In *Proceedings of the 36^{th} ACM Symposium on Theory of Computing (STOC 2004)*, ACM, Chicago, IL, (2004).
19. R. Panigrahy and S. Vishwanathan, An $o(\log^* n)$ approximation algorithm for the asymmetric p-center problem, *Journal of Algorithms* **27** (2), 259–268, (1998).
20. A.F. Archer, Two $o(\log^* k)$-approximation algorithms for the asymmetric k-center problem, In *Proceedings of the Eighth Conference on Integer Programming and Combinatorial Optimization*, pp. 1–14, (2001).
21. R. Raz and S. Safra, A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP, In *Proceedings of the 29^{th} Annual ACM Symposium on Theory of Computing*, pp. 475–484, ACM, (1997).
22. D.S. Johnson, Approximation algorithms for combinatorial problems, *J. Comput. System Sci.* **9**, 256–278, (1974).
23. L.A. Rossman, The EPANET programmer's toolkit for analysis of water distribution systems, In *Proceedings of the Annual Water Resources Planning and Management Conference*, (1999).