Efficient Sensor Placement Optimization for Securing Large Water Distribution Networks

Andreas Krause [*] Jure Leskovec [†] Carlos Guestrin[‡]

Jeanne VanBriesen[§] Christos Faloutsos[¶]

Abstract

We consider the problem of deploying sensors in a large water distribution network, in order to detect the malicious introduction of contaminants. We show that a large class of realistic objective functions – such as reduction of detection time and the population protected from consuming contaminated water – exhibit an important diminishing returns effect called *submodularity*. We exploit the submodularity of these objectives in order to design efficient placement algorithms with provable performance guarantees. Our algorithms do *not* rely on mixed integer programming, and scale well to networks of arbitrary size. The problem instances considered in our approach are orders of magnitude (a factor of 72) larger than the largest problems solved in the literature. We show how our method can be extended to multicriteria optimization,

[*]Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA; E-Mail: krausea@cs.cmu.edu

[†]Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA; E-Mail: jure@cs.cmu.edu

[‡]Machine Learning Department and Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA; E-Mail: guestrin@cs.cmu.edu

[§]Department of Civil & Environmental Engineering and Department of Biomedical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA; E-Mail: jeanne@cmu.edu; ASCE member number 362477

[¶]Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA; E-Mail: christos@cs.cmu.edu

selecting placements robust to sensor failures and optimizing minimax criteria. We provide extensive empirical evidence on the effectiveness of our method on two benchmark distribution networks, and an actual drinking water distribution system of greater than 21,000 nodes.

Keywords: Water distribution networks, contamination detection, optimization, algorithms.

Introduction

Accidental or malicious introduction of a contaminant into water distribution systems could potentially have severe health effects on a population, as well as social and economic impacts. Such intrusions can potentially be detected by deploying a number of sensing devices into the water distribution system. Instrumenting every node of the network is prohibitively expensive, and hence the optimal placement of the sensing devices becomes a crucial issue. For large water distribution networks with tens of thousands up to millions of nodes, solving this optimization becomes a difficult computational challenge.

To catalyze the development of new approaches, the *Battle of Water Sensor Networks* (BWSN) challenge was organized by Ostfeld et.al. during the Water Distribution Systems Analysis Symposium 2006. In this challenge, sensor placements were to be designed for two realistic water distribution networks, and several intrusion settings as formalized in Ostfeld *et al.* (2008). The contributed sensor placements were evaluated with respect to four realistic objective functions – the time until an intrusion is detected (called Z_1), the expected population affected by an intrusion (Z_2), the expected amount of contaminated water consumed (Z_3) and likelihood of detection (Z_4).

In this paper, we present our contribution to this challenge, along with several extensions. Our approach is based on the key observation that the benefit of placing sensors, evaluated according to the BWSN objectives, satisfies *submodularity*, an intuitive diminishing returns property. We use this property to develop fast algorithms with provable guarantees which can handle all settings defined in BWSN, and actually scale far beyond the problem instances defined in this challenge. The main advantages of our approach include:

- A highly efficient algorithm, which scales to networks of tens of thousands of nodes and beyond, and to millions of possible intrusion scenarios.
- We prove rigorous theoretical worst-case bounds about the performance of our algorithm: The solutions produced are guaranteed to be within 63% of the optimal solution, within computational time proportional to the number of nodes and scenarios considered.
- We can also compute online (problem instance dependent) bounds that show that our solutions are usually within 95% of the optimal solution. These online bounds can be applied to the sensor placements returned by *any* algorithm for BWSN.
- Our method naturally extends to the multicriterion optimization setting, and can optimize both average case (where intrusions are selected at random) and worst case placement scores (where intrusions are adversarially chosen after the sensors have been deployed).
- Using our preprocessing, we can exhaustively simulate all 3.6 million intrusion scenarios defined for the large BWSN network, and include them in our optimization. The problem instances considered in our approach are a factor of 72 larger than

the largest problem instances reported in the literature, including those reported by others in the BWSN competition.

- At BWSN, our approach obtained the highest number of non-dominated solutions.

Related work

A large number of approaches have been proposed for optimizing water sensor networks (*c.f.,* Berry *et al.* 2006b, Berry *et al.* 2005, Kessler *et al.* 1998, Kumar *et al.* 1997, Watson *et al.* 2004, Ostfeld and Salomons 2006, Wu and Walski 2006, Dorini *et al.* 2006, Guan *et al.* 2006, Berry *et al.* 2006a, Huang *et al.* 2006, Preis and Ostfeld 2006). Berry *et al.* (2006b) present a concise overview of the prior literature on optimizing sensor networks for contaminant detection. Most of these approaches are only applicable to small networks up to approximately 500 nodes. Many approaches are based on heuristics that cannot provide provable performance guarantees of the solutions.

Closest to ours is an approach by Berry *et al.* (2006b), who equate the placement problem with a p-median problem. They consider two algorithms, one based on Mixed Integer Programming (MIP), and a fast heuristic (GRASP). The MIP approach has high memory consumption, so certain approximations had to be made to solve the BWSN challenge on current hardware (like a coarser water reporting step, and reduced number of contamination scenarios). Furthermore, due to the problem complexity, the approach cannot be expected to find the optimal solution in polynomial time in general. The GRASP heuristic, while faster, does not provide offline guarantees on the solution quality (although LP-relaxations are used to get online bounds).

In contrast, our approach can handle networks of the same size (greater than 12,000 nodes), without requiring the same approximations (e.g., allowing 288 intrusion time steps

per node compared to 4, and 5 minute water quality reporting step compared to 1 hour on comparable hardware). By exploiting the concept of *submodularity*, our approach can compute online bounds on the quality of the sensor network deployment (in our experiments, usually scores provably within 95% of the optimal score where achieved). Additionally, our approach is the first efficient approximation algorithm for the water network placement problem: It is guaranteed to efficiently provide solutions which are at least within 63% of the optimal solution.

Sensor placement objectives

In order to optimize sensor placements, we need to quantify the benefit of a sensor network deployment in reducing the adverse effects of malicious introductions of contaminants into the municipal water network. Several quantitative criteria have been proposed in the past, including volume of contaminated water consumed (Kessler *et al.*, 1998), population affected (Berry *et al.*, 2005) and the time to detection (Kumar *et al.*, 1997). In our work, as a case study, we use criteria similar to those defined by the *Battle of Water Sensor Networks* challenge (Ostfeld *et al.*, 2008). These criteria include the expected time to detect an intrusion (Z_1), the expected total population affected by an intrusion prior to detection (Z_2), the expected total amount of contaminated water consumed prior to detection (Z_3), the likelihood of detecting an intrusion (Z_4). Our approach however can handle a large class of realistic criteria beyond those defined in BWSN.

An *intrusion* is defined by the introduction of a contaminant at a specified point in *time*, for a certain attack *duration*, at a certain *intrusion node*. To describe our approach more formally, consider a set of possible intrusion scenarios $\mathcal{I}$. Each scenario $i \in \mathcal{I}$ is

parameterized by intrusion node, time and duration as defined above. We assume there is a probability distribution P over the possible scenarios, i.e., $P(i)$ is the probability that scenario i occurs. In our experiments we use the uniform distribution, but our approach can handle arbitrary parametrization of the scenario likelihood. We also assume that there is a set of possible *sensor locations* $\mathcal{S}$ we can choose from (e.g., all junctions in the network). A *sensor placement* $\mathcal{A} \subseteq \mathcal{S}$ is a subset of all possible sensor locations. With each sensor $s \in \mathcal{S}$ we associate a nonnegative cost $c(s)$. The cost of a placement $c(\mathcal{A})$ is simply the sum of the cost of the sensors, i.e., $c(\mathcal{A}) = \sum_{s \in \mathcal{A}} c(s)$. For each scenario $i \in \mathcal{I}$ and sensor $s \in \mathcal{S}$ we define the *detection time* $T(s, i)$ as the time it takes for sensor s to detect the intrusion defined by scenario i. If sensor s never detects intrusion i, we set $T(s, i) = \infty$. We can define the detection time for a sensor placement $\mathcal{A}$ and scenario i as $T(\mathcal{A}, i) = \min_{s \in \mathcal{A}} T(s, i)$. For each scenario i, we need to specify the effect of detecting scenario i at time t. To do this, let π_i be a *penalty function*, i.e., $\pi_i(t)$ is the penalty incurred by detecting intrusion scenario i at time t ($t = \infty$ is allowed). Our notion of penalties implies that upon detection, immediate action is taken and no more contaminated water is consumed. This notion can be easily relaxed, e.g., handling the case where the response is delayed by a fixed amount of time. Intuitively, the later an attack is detected, the worse. Therefore we require π_i to be nondecreasing, i.e, for times $t < t'$ and an arbitrary intrusion i, it must hold that $\pi_i(t) \leq \pi_i(t')$. For example, $\pi_i(t)$ can measure the expected amount of population affected by intrusion i at time t. $\pi_i(\infty)$ is the maximum penalty incurred if the scenario i is not detected at all. Using the concept of penalties, we can define the *penalty reduction* for an attack scenario i as $R(\mathcal{A}, i) = \pi_i(\infty) - \pi_i(T(\mathcal{A}, i))$. Since we do not know where the attack will be, but have a probability distribution over possible scenarios, we optimize the

expected penalty reduction $R(\mathcal{A})$ of placement $\mathcal{A}$ given by $R(\mathcal{A}) = \sum_{i \in \mathcal{I}} P(i) R(\mathcal{A}, i)$. Note that any placement maximizing the expected penalty reduction equivalently minimizes the expected penalty $\sum_i P(i)\pi_i(T(\mathcal{A}, i))$.

Example. All four objective functions described above can be written in this form, with appropriate choice of penalty functions. For example, if we want to minimize the time to detection, we would choose $\pi_i(t) = t$, and $\pi_i(\infty) = t_{\max}$, where $t_{\max}$ is a large number, e.g., the number of time steps until the end of the simulation. The *expected penalty reduction* $R(\mathcal{A})$ is then the reduction in detection time, averaged over all scenarios. If we want to model more complex penalties, such as the amount of contaminated water consumed, we would, for scenario i, run an EPANET 2.0 simulation, and, for each simulation timestep t, compute the contaminant concentration at every node. Multiplying with the demand at every node and summing up, we would get the penalty $\pi_i(t)$. For $\pi_i(\infty)$, we can choose the amount of contaminant consumed at the end of the simulation. In Leskovec *et al.* (2007), we describe how our formalism applies to a very different application domain, and how it can be used to select informative weblogs on the Internet.

Properties of the penalty reduction. The expected penalty reduction has several intuitive properties. It is nonnegative (i.e., $R(\mathcal{A}) \geq 0$) for all placements $\mathcal{A}$, and we generally want to maximize the penalty reduction. $R(\emptyset) = 0$, i.e., if we place no sensors, the penalty reduction is 0. We can also see that R is *nondecreasing*, i.e., for subsets $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{S}$, it holds that $R(\mathcal{A}) \leq R(\mathcal{B})$, hence the penalty can only decrease if we place more sensors. There is an additional intuitive property: If we add a sensor to a large deployment, we would expect less penalty reduction than if we add the sensor to a small deployment. This *diminishing returns* is formalized by the combinatorial concept of *submodularity* (*c.f.,*

Nemhauser *et al.* 1978): A set function F is called submodular if for all subsets $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{S}$ and elements $s \in \mathcal{S}$ it holds that $F(\mathcal{A} \cup \{s\}) - F(\mathcal{A}) \geq F(\mathcal{B} \cup \{s\}) - F(\mathcal{B})$, i.e., adding s to the smaller set $\mathcal{A}$ helps more than adding it to the larger set $\mathcal{B}$. In fact, we can prove that *all* penalty reduction functions as defined above are submodular:

Theorem 1. *The penalty reduction function R is submodular.*

The proof of this Theorem is given in the Appendix. Thus, the optimization of sensor placements for water distribution networks can be cast as a submodular optimization problem. The penalty reduction objective R is similar to one of the examples of submodular functions described by Nemhauser *et al.* (1978). Our objective, however, preserves additional problem structure (sparsity) which we exploit in our implementation, and which we crucially depend on to solve large problem instances. We want to maximize the penalty reduction, subject to a constraint on the cost of the placement, i.e., solve $\max_{\mathcal{A}:c(\mathcal{A})\leq B} R(\mathcal{A})$. Hereby, B is a *budget* we can spend for deploying sensors. Note that $R(\mathcal{A})$ could be difficult to compute for any given placement, as for large networks we have to perform a large number of simulations in order to accurately evaluate $R(\mathcal{A})$. We also have to search through an exponential number of possible placements $\mathcal{A}$, which is not practical for large real-world networks.

One can show that the optimization problem is computationally complex (NP-hard), such that, even though in certain cases the optimal solution can be found efficiently (Berry *et al.*, 2006a), in general we cannot expect to find the optimal solution in reasonable time (Garey and Johnson, 2003). A key result of our work is that by connecting monitoring water networks and submodular function maximization, we can exploit existing algorithms for optimizing submodular functions, leading to strong guarantees about the quality of the

obtained solutions.

Multicriteria optimization

Often, we are interested in trading off several objective functions, for example, the affected population and the detection likelihood. In this case, we have several objective functions $R_1, \ldots, R_m$. In general, we cannot expect to find a placement that outperforms all other placements in all criteria. Hence we will be interested in *Pareto-optimal* placements: A placement $\mathcal{A}$ is called Pareto-optimal if it is not dominated, i.e., if there does not exist another placement $\mathcal{B}$ such that $R_j(\mathcal{B}) \geq R_j(\mathcal{A})$ for all $1 \leq j \leq m$, whereby at least one inequality is strict (i.e., $\mathcal{B}$ is better than $\mathcal{A}$ in at least one criterion, and at least as good in all the others). In order to find Pareto-optimal placements, one can use *scalarization* (Boyd and Vandenberghe, 2004). Let $\lambda_1, \ldots, \lambda_m$ be positive real numbers. Then the solution to the problem $\max_{\mathcal{A}} \sum_j \lambda_j R_j(\mathcal{A})$, subject to $c(\mathcal{A}) \leq B$ is Pareto optimal. Different choices of the weights λ_j will give us different Pareto optimal solutions. In fact, since nonnegative linear combinations of submodular functions are submodular, the new objective $R(\mathcal{A}) = \sum_j \lambda_j R_j(\mathcal{A})$ is again submodular. Hence, solutions obtained by scalarization provide us with rigorous bounds on the location of the true Pareto optimal surface. To facilitate comparison and simplify the interpretation of the weights λ_i, we normalize the R_i. Let $\bar{\pi}_j = \sum_{i \in \mathcal{I}} P(i) \pi_i^{(j)}(\infty)$, i.e., $\bar{\pi}_j$ is the average penalty incurred for not detecting any scenario. Then $R'_j(\mathcal{A}) = R_j(\mathcal{A})/\bar{\pi}_j$ is normalized between 0 and 1, where the optimum, 1, is attained if all intrusions are detected early enough such that no penalty is incurred for any scenario.

Algorithms for optimization

Above, we have shown that a large class of important sensor placement objectives (reduction in detection time and population affected, etc.) is submodular and nondecreasing. Let us first consider the case where each sensor has equal cost, and we can place k sensors. There are $\binom{|\mathcal{S}|}{k}$ possible placements, in general far too many to be searched exhaustively. For this reason, many heuristic approaches have been developed, which try to find good solutions through a search procedure (e.g., genetic algorithms (Guan *et al.*, 2006; Ostfeld and Salomons, 2004; Wu and Walski, 2006; Preis and Ostfeld, 2006), cross-entropy selection (Dorini *et al.*, 2006), predator-prey heuristics (Gueli, 2006), etc.). We are not aware of any existing method that can provide provable guarantees about the performance of the search procedure. The only exception are the mixed integer programming solutions (*c.f.,* (Berry *et al.*, 2005, 2006a; Propato and Piller, 2006)), which provide bounds (or compute the optimal solution), but which do *not* have any runtime guarantees. Since sensor deployments are expensive, we want to find an optimization procedure, which finds provably near-optimal sensor placements. In the following, we present several algorithms that have such strong theoretical guarantees.

The greedy algorithm

A natural approach to finding approximate solutions is the *greedy algorithm*: This algorithm starts with the empty placement, $\mathcal{A} = \emptyset$, and proceeds iteratively. At the j-th round, it selects the sensor $s = \text{argmax}_{s \in \mathcal{S}} R(\mathcal{A} \cup \{s\}) - R(\mathcal{A})$, i.e., the sensor which will decrease the expected penalty the most, and adds it to the current set, $\mathcal{A} \leftarrow \mathcal{A} \cup \{s\}$. Perhaps surprisingly, a fundamental result by Nemhauser *et al.* (1978) shows that this in-

tuitive procedure is *near-optimal* for the class of nondecreasing submodular functions: The greedy algorithm always picks a set $\mathcal{A}_G$ such that $R(\mathcal{A}_G) \geq (1-1/e)R(\mathcal{A}^*)$, where $\mathcal{A}^*$ is the optimal placement of k sensors. Hence, the greedy solutions achieve a penalty reduction of at least $1 - 1/e \approx 63\%$ times the optimal penalty reduction over all placements of size k. The greedy algorithm has been considered previously by Uber *et al.* (2004), albeit without providing theoretical guarantees. The running time of the algorithm is proportional to the number of locations $|\mathcal{S}|$, the number of sensors to be placed k and the time T_{eval} it takes to evaluate $R(\mathcal{A})$ for any placement $\mathcal{A}$. More formally, the running time is $\mathcal{O}(k \cdot |\mathcal{S}| \cdot T_{eval})$.

Location-specific placement costs

There are many realistic scenarios in which different sensor locations might have a different cost. For example, certain locations in the network might be more expensive to instrument than others because they are less accessible. We might also have several sensor models to choose from, e.g., a high cost-high accuracy sensor, and a low-cost low-accuracy sensor. When sensing locations can have different costs, the simple greedy algorithm does not have theoretical quality guarantees. However, a slightly more complex algorithm, combining the greedy algorithm with partial enumeration, also achieves a $1-1/e$ approximation guarantee by exploiting submodularity (Sviridenko, 2004; Krause and Guestrin, 2005).

Online bounds

The 63% of the optimal bound on the greedy algorithm is an *offline bound*, i.e., we guarantee this performance even before running the algorithm. In addition, the submodularity of the penalty reduction R allows us to compute *online bounds* on the quality of our solution, i.e., once we obtain a solution, we can provide an often much tighter bound on its quality.

These online bounds apply to the solution obtained by the greedy algorithm, or by any other algorithm for optimizing sensor placements. Let $\mathcal{A}$ be a solution, and for each sensor location s which is not in $\mathcal{A}$, let $\delta_s = R(\mathcal{A} \cup \{s\}) - R(\mathcal{A})$ be the improvement in penalty reduction we would get by adding a sensor at location s. Assume we want to place k sensors. Let $s_1, \ldots, s_k$ be the k locations for which δ_s is largest. Then it holds that $R(\mathcal{A}^*) \leq R(\mathcal{A}) + \sum_{j=1}^{k} \delta_{s_j}$. This bound directly follows from the submodularity of R, since $R(\mathcal{A}^*) \leq R(\mathcal{A} \cup \mathcal{A}^*) \leq R(\mathcal{A}) + \sum_{s \in \mathcal{A}^*} \delta_s \leq R(\mathcal{A}) + \sum_{j=1}^{k} \delta_{s_j}$, and it allows us to get guarantees about *arbitrary* placements $\mathcal{A}$.

Mixed integer programming

The submodularity of R also allows us to adopt a mixed-integer programming approach developed by Nemhauser and Wolsey (1981). The mixed integer program (MIP) is given by:

$$
\begin{aligned}
&\max \eta; \\
&\quad \eta \;\leq\; R(\mathcal{B}) + \sum_{s_i \in \mathcal{S} \setminus \mathcal{B}} \alpha_i [R(\mathcal{B} \cup s_i) - R(\mathcal{B})], \quad \forall \mathcal{B} \subseteq \mathcal{S}; && (1) \\
&\sum_i \alpha_i \;\leq\; k, \quad \forall i; && (2) \\
&\quad \alpha_i \;\in\; \{0, 1\}, \quad \forall i;
\end{aligned}
$$

where $\alpha_i = 1$ means that location s_i should be selected. Note that this MIP can also handle the case in which each location can have a different cost, by replacing the constraint (2) by $\sum_i \alpha_i c_i \leq B$, where B is the budget and $c_i = c(s_i)$.

Unfortunately, this MIP has exponentially many constraints of type (1). Nemhauser and Wolsey (1981) proposed the following constraint generation algorithm: Let $\bar{\alpha}^{\mathcal{A}}$ denote an assignment to $\alpha_1, \ldots, \alpha_n$ such that $\alpha_i = 1$ iff $s_i \in \mathcal{A}$. Starting with no constraints of type (1), the MIP is solved, and one checks whether the current solution $(\eta, \bar{\alpha}^{\mathcal{B}})$ satisfies

$\eta \leq R(\mathcal{B})$). If the solution does not, a violated constraint has been found. Since solving individual MIP instances (even with only a small number of constraints) is a complex (NP-hard) problem, we need to resort to search heuristics such as Branch and Bound and Cut during the constraint generation process.

Improving solutions through search

If solving the optimization problem using mixed integer programming is intractable, we can run the greedy algorithm to find a (provably) good initial solution, and improve on this solution using local search. Any existing sensor placement technique can be used here. In our experiments, we used a variant of simulated annealing (Kirkpatrick *et al.*, 1983). This stochastic approach, in each round, proposes an exchange of a selected location $s \in A$ and an unselected location $s' \notin A$. The algorithm then computes the difference in scores $\delta = R(\mathcal{A} \cup \{s'\} \setminus \{s\}) - R(\mathcal{A})$. If this difference is positive (i.e., the new solution has a higher score), the proposal is accepted. If the difference is negative, the proposal is accepted with probability $\exp(\delta/\vartheta_t)$, where ϑ_t is the *annealing temperature* at round t. We use a harmonic annealing schedule, where $\vartheta_t = C/t$, for some constant C. Intuitively, large temperatures allow exploring new sensor locations, which might locally decrease the score R, but might lead out of a local minimum. The more rounds the algorithm goes through, the smaller ϑ_t gets, and eventually, proposals that decrease the score become very unlikely. Here, our online bounds can again be used to provide bounds on the quality of the solution obtained.

A note on minimizing the expected penalty

While maximizing the expected penalty reduction is equivalent to minimizing the expected penalty, the $(1 - 1/e)$ offline guarantee does not transfer from one setting to the other.

More formally, we have the following result:

Theorem 2. *Unless P=NP, no polynomial time algorithm can give any offline approximation guarantee on the penalty minimization problem.*

It is widely believed that P is not equal to NP (Garey and Johnson, 2003). Thus it is unlikely that any algorithm can be developed to efficiently minimize *penalties* directly, further motivating our focus on *penalty reduction maximization*. Nonetheless, any online bounds we obtain for the penalty reduction case can be turned into bounds for the penalty minimization problem. For example, suppose we have an approximate solution $\mathcal{A}'$, and that the optimal *penalty reduction* OPT is upper-bounded by $R(\mathcal{A}') \leq OPT \leq M$ for some value M. Then we know that the optimal *penalty* π^* is lower-bounded by $\bar{\pi} - M$, where $\bar{\pi}$ is the expected penalty if no sensors are placed. We illustrate the usefulness of this observation in our experimental results.

Extensions

Adversarial objective functions

The definition of our penalty reduction is an *average case* objective. We believe that the likelihood of any particular intrusion i is specified by its probability $P(i)$, and $R(\mathcal{A}) = \sum_{i \in \mathcal{I}} P(i) R(\mathcal{A}, i)$.

Instead of optimizing this average case performance, we might be interested in the *worst case* performance. Here, we would define $R_w(\mathcal{A}) = \min_{i \in \mathcal{I}} R(\mathcal{A}, i)$; the score of a placement is the minimum penalty reduction over all scenarios. Unfortunately, $R_w(\mathcal{A})$ is not a submodular function in general.

In order to optimize this objective, we can perform the following transformation. Instead of using the penalty reduction $R(s, i)$ directly, we apply the monotonic transfor-

mation, $R'(s,i) = 1 - \exp(-R(s,i)/\vartheta)$, for some positive real number ϑ, and then optimize $R'(\mathcal{A}) = \sum_i P(i)R'(s,i)$ instead of the minimum. After transformation, $R'(\mathcal{A}) = \sum_i P(i)R'(s,i)$ is again a submodular function, and all proposed techniques apply. In order to understand why this transformation makes sense, we can look at the derivative of the function $g(x) = 1 - \exp(-x/t)$, $\frac{\partial}{\partial x}g(x) = \exp(-x/t)/t$. The smaller x, the more any improvement δ helps. By placing a sensor, some of the scenario specific penalty reductions $R(\mathcal{A}, i)$ increase. After transformation, this increase is largest for scenarios for which $R(\mathcal{A}, i)$ is smallest. Hence the algorithm has an incentive to increase the penalty reduction of the worst case scenario. Hence, when optimizing the transformed criterion, a good placement will try to uniformly increase the penalty reductions over all scenarios. This approach effectively approximates the non-submodular adversarial score by a submodular set function. In very recent work, we have developed an algorithm for directly optimizing adversarial scores, such as $R_w(\mathcal{A}) = \min_i R(s,i)$, (Krause *et al.*, 2007); the application of this new approach to water distribution systems will be the focus of future work.

Robustness of sensor placements

As with any physical device, sensor nodes are susceptible to failures. Loss of power for example could stop a sensor from making further measurements. Additionally, especially if the contaminant concentration is low, the sensor might not detect an intrusion. Our approach can be extended to handle such failures. With each location $s \in \mathcal{S}$, we associate a discrete random variable F_s such that $F_s = 0$ indicates that a sensor placed at location s has failed and will not produce any measurements, and $F_s = 1$ indicates that the sensor is working correctly. Similarly, we could use a continuous random variable F_s which models the minimum contaminant concentration required for detecting a contamination at location

s. For a placement $\mathcal{A} \subset \mathcal{S}$, denote by $\mathcal{A}_\mathbf{f}$ the subset of locations $s \in \mathcal{A}$ such that $F_s = 1$, or, respectively, the subset of locations where the contaminant concentration exceeds the minimum concentration specified by F_s. Hence $\mathcal{A}_\mathbf{f}$ denotes the subset of functional sensors. Then, the robust penalty reduction $R(\mathcal{A}) = \mathbb{E}_\mathbf{F}[\mathcal{A}_\mathbf{f}] = \sum_\mathbf{f} P(\mathbf{f}) R(\mathcal{A}_\mathbf{f})$, is an expectation of the penalty reduction achieved for placement $\mathcal{A}$ where all possible failure scenarios are considered. Since the class of submodular functions is closed under non-negative linear combinations, we can see that $R(\mathcal{A})$ is again a nondecreasing submodular function.

Unfortunately, the number of possible failure scenarios grows exponentially in $|\mathcal{S}|$. However, if the F_s are independent and identically distributed, and if the failure probability $P(F_s = 0) = \theta$ is low enough, R can be approximated well, for example, by only taking into account scenarios where none or at most one sensor fails. This simplification often works in practice (Lerner and Parr, 2001).

System Implementation

In order to implement any of the algorithms discussed above, we need to evaluate the penalty reduction function R. Looking at the definition, $R(\mathcal{A}) = \sum_{i \in \mathcal{I}} P(i) R(\mathcal{A}, i) = \sum_{i \in \mathcal{I}} P(i)[\pi_i(\infty) - \pi_i(T(\mathcal{A}, i))]$, we need to compute a sum over all possible intrusion scenarios, and evaluate the time to detection T and the corresponding penalty π_i for each scenario i and each possible set of sensors $\mathcal{A}$. The water distribution networks we considered had 129, 12,527 and around 21,000 nodes, where a sensor could be possibly placed, and where an intrusion could potentially happen. In order to compute the penalties, we perform water quality simulations using the EPANET 2.0 software (Rossman, 1999), with a temporal resolution of 5 minutes. We run our simulations for a 48 hour period, which amounts to 576 water quality time steps. We assume that an intrusion can happen

at an arbitrary point in time within the first 24 hours, which amounts to 288 different starting times for an intrusion at every node in the network. Hence, the total number of intrusion scenarios as defined in the BWSN challenge amounts to 3.6 million. If we consider storing the contaminant concentration for every scenario, for every node and every simulation time step, we need to process a volume of roughly 47 Terabytes of data, just to evaluate a single sensor placement. Considering this amount of data, one might conclude that it is necessary to subsample the number of simulated scenarios (Ostfeld and Salomons, 2006; Wu and Walski, 2006), or decrease the temporal or spatial resolution of the simulations (Berry *et al.*, 2006b; Dorini *et al.*, 2006; Guan *et al.*, 2006). However, these simplifications decrease the accuracy of the computed score, and incur variance (and hence uncertainty) in the prediction, which is undesirable in a critical application such as securing water networks. Also, one might conclude that optimizing even larger water distribution networks is intractable for current computers. In the following, we present how we were able to reduce the amount of data by several orders of magnitude (from 47 Terabytes to 16 Gigabytes) without losing any information, thereby reducing the evaluation time T_{eval} of the penalty reduction R over all scenarios for any given placement to fractions of a second.

Fast computation of the score function

Considering the running time of an EPANET 2.0 simulation on our network (roughly 4 seconds on a current Pentium 4 3GHz), the time required for exhaustive simulation of all 3.6 million scenarios would require roughly 170 days. The simulations however are easily distributable on a cluster of several machines, where each machine performs a subset of the simulations. Using 20 ordinary desktop computers, the computation time reduces to roughly 9 days, which is very small compared to the time required for deploying a

water sensor network. Even though running all simulations once is realistic, running all simulations for every network design we want to evaluate is intractable. Hence, we need to store enough information during our single run of the simulations, such that we are able to evaluate the score $R(\mathcal{A})$ for every placement $\mathcal{A}$ very quickly. Storing all 47 Terabytes of data is prohibitive, especially since the entire data set needs to be scanned once in order to evaluate each placement. Looking at the definition of R, we can conclude that the *only* information we need is the following: (i) For each potential sensor location $s \in \mathcal{S}$ and intrusion scenario $i \in \mathcal{I}$, we need the time to detection $T(s, i)$, and (ii), for each scenario $i \in \mathcal{I}$, we need the maximum penalty $\pi_i(\infty)$, and the penalty for each detection time, $\pi_i(t)$.

The data required to store $T(s, i)$ is at most 84 Gigabytes, and the information to store π_i is at most 8 Gigabytes. In our experiments we also noticed that most scenarios are only detected by very few sensor locations, hence $T(s, i)$ is a very sparse matrix. Using an appropriate sparse representation, the relevant results from all simulations required roughly 16.3 Gigabytes. Note that in order to exploit this sparsity, the interpretation of scores as penalty reductions is crucial, since this interpretation allows us to ignore all undetected scenarios (as their contribution to the score is 0). Hence, by exploiting problem structure (penalty reduction representation and sparsity), we were able to reduce the original data footprint of 47 Terabytes down to 16 Gigabytes, without losing any information. This is small enough that our HP 64-bit server with 32 Gigabytes of main memory can have access to all the relevant information *without* having to rely on harddisk accesses. Since memory accesses are several orders of magnitudes faster than harddisk accesses, this greatly speeds up computation.

The following describes how we quickly compute the score R for arbitrary sensor place-

ments $\mathcal{A}$.

1. For each sensor $s \in \mathcal{A}$, retrieve the set of detected scenarios $\mathcal{B}$, and for each detected scenario i, compute the penalty reduction $R(s,i) = \pi_i(\infty) - \pi_i(T(s,i))$.

2. For each scenario i, compute the overall penalty reduction $R(\mathcal{A},i) = \max_{s\in\mathcal{A}} R(s,i)$.

3. Sum up all penalty reductions $R(\mathcal{A}) = \sum_i P(i)R(\mathcal{A},i)$.

Notice that the sum in step 3 is only over the detected scenarios (since the penalty reduction of all undetected scenarios is 0 by definition). Other researchers (*c.f.,* Dorini *et al.* 2006 and Berry *et al.* 2006b) have used similar data structures and preprocessing procedures.

Fast implementation of the greedy algorithm

Even if we can quickly evaluate the score $R(\mathcal{A})$ of any given placement, we still need to perform a large number of these evaluations in order to run the greedy algorithm. If we select k sensors among n locations, we roughly need kn function evaluations. We can use a computational trick to require far fewer function evaluations in practice. Assume we have computed the greedy improvements $\delta_s = R(\mathcal{A}\cup\{s\}) - R(\mathcal{A})$ for all $s \in \mathcal{S}\setminus\mathcal{A}$. The key idea is to realize that adding a sensor s' to a placement $\mathcal{A}$ often does not change the scores δ_s for many sensor locations s, and – more importantly – can never increase any scores δ_s due to submodularity of $R(\mathcal{A})$. So instead of recomputing δ_s for every sensor after adding s' (and hence requiring $n-|\mathcal{A}|$ evaluations of R), we perform *lazy* evaluations: Initially, we mark all δ_s as *invalid*. When finding the next location to place a sensor, we go through the locations in decreasing order of their δ_s scores. If the δ_s for the top location s is marked as invalid, we recompute it, and insert it into the existing order of the δ_s. In many cases,

the recomputation of δ_s will lead to a new value which is not much smaller, and hence often, the top element will stay the top element even after recomputation. In this case, we found a new sensor to add, without having reevaluated δ_s for every location s. This lazy procedure can be shown to be correct due to the submodularity of R, and leads to far fewer evaluations of R. In our experiments, we often achieve a factor 30 improvement in speed when placing 20 sensors, hence allowing us to run the greedy algorithm within an hour on an Intel Xeon 3GHz processor.

Results

Networks analyzed

We considered both the small network on 129 nodes (BWSN1), and a large, realistic, 12,527 node distribution network (BWSN2) provided as part of the BWSN challenge (Ostfeld *et al.*, 2008). In addition to the two networks defined by the BWSN challenge, we consider a third water distribution network (NW3) of a large metropolitan area in the United States. The network (not including the household level) contains around 21,000 nodes and 25,000 pipes. To our knowledge, this is the largest water distribution network considered for sensor placement optimization so far. Due to security concerns, we cannot provide actual placements for this network. As the purpose of the experiments is to show the scalability of our method, we will mainly provide running time analyses.

Impact analysis

In the small network (BWSN1), there are a total of 37,152 intrusion scenarios, and for the large network (BWSN2), a total of 3.6 million scenarios. Among these scenarios, one would expect some of them to have significant effect on the network, whereas in other scenarios,

only a few nodes would be affected. In order to understand this variability, we computed a histogram (for BWSN2): Fig. 1 shows the frequency of scenarios that affect a specific number of nodes in the network. We can see that the vast majority of scenarios are highly localized, i.e., only affects a small number of nodes. However, the distribution is heavy tailed: There are a few scenarios that affect a large part of the network. These are the scenarios that are potentially most dangerous, and network deployments should take these scenarios into account. Thus, since the number of scenarios that affect the entire network is small, a significant limitation of methods that randomly pre-select a smaller number of intrusion scenarios on which to base optimization is that a small random subset of intrusion scenarios is not likely to contain these high impact scenarios.

Single objective optimization

We performed several experiments in order to analyze the performance of our placement algorithms. We first optimized placements using the greedy algorithm. Figures 2a and 2b show the diminishing returns effect when optimizing each of the BWSN objective functions on the small (BWSN1) and large (BWSN2) network. They show the penalty reductions (scores) achieved when using the greedy algorithm to optimize the different criteria (e.g., time to detection). Interestingly, the behavior is very different, depending on which function we optimize. For (BWSN1), when trying to minimize the contaminated water consumed (Z_3), only two sensors suffice to achieve a score very close to 1, the optimal penalty reduction. When optimizing the time to detection (Z_1), however, the score increases very quickly at the beginning, but continues to slowly increase later on. Intuitively, this is the case because no matter where the contaminant is introduced, it spreads rather slowly across the network. In order to achieve instantaneous detection, a very dense deployment

is required. Also the plot shows that the detection likelihood (Z_4) achieves its maximum already when placing 15 sensors. Note that this maximum value is less than 1, as some intrusion scenarios *never* have an effect on the network, i.e., never lead to a nonzero contaminant concentration at any node in the network, as the contaminant is drained from the network instantaneously. For (BWSN2), the contaminated water consumed (Z_3) score flattens out most quickly, similarly to the previous experiment. The time to detection (Z_1) achieves the lowest penalty reduction, and keeps increasing roughly linearly after 10 sensors have been placed. The explanation for this effect is that there are a few high impact scenarios that need to be detected in order to achieve high scores for Z_2 and Z_3. Most scenarios are very local and not detected by most sensor locations, hence the detection likelihood (Z_4) and time to detection score (Z_3) stay very low, unless a large number of sensors is placed. Tab. 1 compares memory use and run time on all networks.

Comparison with random placements and heuristics. In order to see how the optimized placements compare against random placements, we selected 100 placements of increasing sizes uniformly at random, and compared them with the scores obtained by the greedy algorithm. Figure 2c presents the results of this experiment for the large network (BWSN2) when optimizing Z_1 (panel a). In addition to the median score, we present minimum, maximum, 10%, 25%, 75% and 90%-iles. Even the maximum over 100 random placements is worse than the greedy solution. We also compared the optimized selection with various heuristics, like selecting the nodes with highest degree (number of pipes connecting to a node), population at a node, average flow and average diameter of connected pipes. Figure 2d presents the results of this comparison for Z_2 on BWSN2. None of the heuristics performs significantly better than random selection. Interestingly,

even though our objective Z_2 attempts to minimize the affected population, simply placing sensors at high population nodes leads to bad performance.

Online and offline bounds. In order to study the approximation quality of the greedy algorithm, we computed both the online and offline bounds on the penalty reduction, as described above. Figure 3 shows the results of this experiment on the large network (BWSN2), when optimizing the time to detection (Z_1) and contaminated water consumed (Z_3). For Z_3 (see Fig. 3b), we can see that the offline bound of $(1 - 1/e)^{-1}$ times the achieved penalty reduction score quickly becomes meaningless when placing more than 3 sensors, since the maximum score attainable is 1. The online bound however quickly becomes very tight as we increase the number of sensors to place. From the online bounds we can see that the greedy algorithm achieves scores, for example, within 95% of optimum, when placing 20 sensors. For Z_1 (see Fig. 3a), we can see that the offline bound does not become meaningless (as the magnitude of the penalty reduction for Z_1 is less than for Z_3). However, the online bound again is much tighter, showing that the greedy solution is within at least 80% of the optimum solution when placing 20 sensors. Note that even this online bound can be loose, and we expect that this greedy solution is significantly better than 80% of optimal.

Mixed Integer Programming and local search. We also used the mixed integer program described above, in order to compute the optimal solution for placing up to 6 sensors in the small network (BWSN1) of 129 nodes, when optimizing time to detection. The MIP was able to find the best placement of 6 sensors within 6 minutes on a Pentium Mobile processor. Our estimates show that exhaustive search for the optimal placement would take approximately 430 days. We also found that for placements up to size 6,

the greedy algorithm actually found the *optimal* solution. Fig. 3c compares the running times for exhaustive search, mixed integer programming, the greedy algorithm and the fast (lazy) greedy algorithm, when optimizing placements up to size 6. We can observe an improvement by orders of magnitude in the sequence of these approaches.

We performed the same experiment on the expected population affected (Z_2). Here, we could not solve the MIP instances (which were harder due to the complex nature of the Z_2 penalty function) exactly, but we still used the MIP to get tight bounds on the optimal solution. Fig. 3d presents the result of the greedy algorithm and the best solution found by simulated annealing with 10,000 iterations (which always was at least as good as the solutions obtained by the MIP solver). We also plot the tightest bound obtained from the MIP within 5 minutes of computation time. The greedy algorithm always found solutions within 98% of optimal, for placements up to size 10. We can convert the normalized penalty reduction back into the expected population affected. For BWSN1, the expected population affected when not placing any sensors is 899.7. After greedily placing 10 sensors, the expected population affected is reduced to 899.7*(1-.8985) = 91.3, since the normalized penalty reduction is .8985. The bounds obtained from the MIP solver guarantee that even under the optimal solution, the expected population affected is at least 73.8.

Multicriteria optimization and results from BWSN

Watson *et al.* (2004) suggested that the objective functions considered (Z_1 through Z_4), are not strongly correlated. We designed an experiment to quantitatively analyze the tradeoff between the objectives by performing multicriteria optimization. In this experiment, we computed the (approximate) Pareto frontiers for trading off pairs of objective functions (see Fig. 4). In order to approximate the Pareto curve, we used different scalarization

parameters. When trading off objective functions R_1 (for Z_1) and R_2 (for Z_2), we optimized $R(\mathcal{A}) = pR_1(\mathcal{A}) + (1-p)R_2(\mathcal{A})$ for various choices of p, such that $0 \leq p \leq 1$. Fig. 4b shows the approximate Pareto frontier for trading off the detection likelihood (Z_4) and the expected population affected (Z_2), for placements of various sizes, on the large network (BWSN2). The tradeoff curves have a *knee*, a point where a small increase in either objective leads to a strong decrease in the other objective. This knee indicates that if we maximize Z_2 we will get a bad score for Z_4 and vice-versa. However, these two objectives are not incompatible: at the knee point we get near-optimal values for both objectives. For example, when placing 20 sensors, by picking a point on the knee, we can detect 42% of all contamination events, while reducing the affected population by 78% (from 1534 to 334). If only optimizing for Z_2, the affected population can be reduced by 83%, but only 34% of the contaminations are detected. If optimizing only for Z_4, 45% of the contaminations are detected, but only 50% reduction of the affected population is achieved. The solution at the knee is only 7% worse with respect to the best Z_2 and Z_4 achievable (using greedy), when only optimizing for Z_2 and Z_4 respectively.

Fig. 4a shows the same experiment, but for trading off the detection likelihood (Z_4) and the expected contaminated water consumed (Z_3). Again, we can see very pronounced knees in the tradeoff curve. We also traded off the expected contaminated water consumed (Z_3) and expected population affected (Z_2). Fig. 4c shows that while there is some variability in the Pareto frontier for very small placements, the Pareto-curves become very dense clusters if we have more than five sensors available. Thus, there is little difference in optimizing for Z_2 or for Z_3. This is not surprising, as we expect correlation between the amount of contaminated water consumed (Z_3) and the population affected (Z_2) by an intrusion, since

both are based on the demand at the nodes.

We submitted our placements with equally weighted objective functions to BWSN, where Ostfeld *et al.* (2008) independently evaluated the contributed solutions of 15 participants. Since many solutions were non-dominated (and hence incomparable), the organizers of BWSN did not select a winner. However, in the conclusion of their analysis, for a collection of comparisons, they counted the number of non-dominated solutions. Our approach achieved the highest number with 26 out of 30 non-dominated solutions. The next best set of placements according to their evaluation using this metric was the one by Berry et al. with 21 out of 30 non-dominated solutions.

Adversarial objectives

We also optimized adversarial scores for the large network (BWSN2). Here, a large number of scenarios affects only a small number of nodes. In fact, more than 2,000 sensors (2,263 in our experiment) would be needed to detect all 3.6 million scenarios. Fig. 5a plots the detection likelihood for an increasing number of optimally chosen sensors. We can see that we need exponentially more sensors to detect the scenarios that have low impact than to detect scenarios that have large impact. This indicates that optimizing the adversarial score is not a reasonable objective for the large network (BWSN2). However, if we make the assumption that the adversary will only choose scenarios which affect a significant part of the network (e.g., at least 500 nodes), 23 sensors suffice to detect all scenarios. Fig. 5b shows that when optimizing the average score, the adversarial score remains 0 even when placing 50 sensors. When optimizing the adversarial score however, it increases to 40% when placing 23 sensors and to 60% when placing 50 sensors. The average score achieved is almost high as when optimizing the average score directly.

Real 21,000 node network

In order to see how our methods scale, we performed experiments on an actual metropolitan area network (NW3), which has more than $21,000$ nodes. We chose the same parameters as in the BWSN challenge. We sampled 150,000 scenarios; on 6 threads in parallel, the simulations completed in 4 days. The compressed data requires approximately 2 GB of memory. Greedily optimizing a set of 30 sensors takes approximately 12 minutes on our 4 GHz HP 64-bit server. Fig. 5c presents the greedy scores obtained when optimizing a scalarization with equal weight on all four objectives, $Z_1, \ldots, Z_4$. The figure also shows the offline bound, and the online bound which becomes quite tight when placing more than 20 sensors. The bounds guarantee that when placing 30 sensors, the achieved solution is within 91% of the optimum score. Fig. 5c also compares the greedy solution with 100 random placements. Hence even the maximum score achieved among 100 random placements is less than 84% of the penalty reduction achieved by the greedy algorithm. Equivalently, one would need to place 21 sensors at random to achieve a median score which is as high as the greedy score for 5 sensors. These results show that our presented methodology is both tractable and applicable to large-scale water distribution sensor placement problems.

Conclusions

We presented an efficient approach towards optimizing sensor placements for securing water distributions against contaminant intrusions. Unlike previous work in this area, our algorithm provides both rigorous approximation *and* running time guarantees. By exploiting submodularity, our approach allows us to compute tight online bounds that can be used to verify the proximity to the optimal solution. Our approach is able to handle placement

problems which are a factor of 72 larger than problems previously considered. Our method also naturally extends to multicriterion optimization and adversarial scoring functions.

Acknowledgements

We would like to thank Stratos Papadomanolakis, Shannon Isovitsch, Jianhua Xu, Mitch Small, Paul Fischbeck, Jared Cohen and the anonymous referees for valuable discussions and feedback. This work was partially supported by NSF Grant No. CNS-0509383, CNS-0625518, SENSOR-0329549, IIS-0534205 and by the Pennsylvania Infrastructure Technology Alliance (PITA), with additional funding from Intel and NTT. Carlos Guestrin was partly supported by an Alfred P. Sloan Fellowship and an IBM Faculty Fellowship. Andreas Krause and Jure Leskovec were partly supported by a Microsoft Research Graduate Fellowship. Most of the experiments were conducted on a HP server sponsored by Hewlett-Packard.

References

Berry, J. W, Fleischer, L, Hart, W. E, Phillips, C. A, and Watson, J. P. Sensor placement in municipal water networks. *J. Water Resources Planning and Management*, 131(3):237–243, 2005.

Berry, J, Hart, W, Phillips, C. A, and Watson, J. P. A facility location approach to sensor placement optimization. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006.

Berry, J, Hart, W. E, Phillips, C. E, Uber, J. G, and Watson, J. Sensor placement in municipal water networks with temporal integer programming models. *J. Water Resources Planning and Management*, 132(4):218–224, 2006.

Boyd, S and Vandenberghe, L. *Convex Optimization*. Cambridge UP, 2004.

Dorini, G, Jonkergouw, P, , Kapelan, Z, Pierro, F di, Khu, S, and Savic, D. An efficient algorithm for sensor placement in water distribution systems. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006.

Garey, M. R and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 2003.

Guan, J, Aral, M. M, Maslia, M. L, and Grayman, W. M. Optimization model and algorithms for design of water sensor placement in water distribution systems. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006.

Gueli, R. Predator-prey model for discrete sensor placement. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006.

Huang, J. J, McBean, E. A, and James, W. Multi-objective optimization for monitoring sensor placement in water distribution systems. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006.

Kessler, A, Ostfeld, A, and Sinai, G. Detecting accidental contaminations in municipal water networks. *J. Water Resources Planning and Management*, 124(4):192–198, 1998.

Kirkpatrick, S, Gelatt, C. D, and Vecchi, M. P. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Krause, A and Guestrin, C. A note on the budgeted maximization of submodular functions. Technical report, CMU-CALD-05-103, 2005.

Krause, A, McMahan, B, Guestrin, C, and Gupta, A. Selecting observations against adversarial objectives. In *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2007.

Kumar, A, Kansal, M. L, and Arora, G. Identification of monitoring stations in water distribution systems. *J. Environmental Engineering*, pages 746–752, 1997.

Lerner, U and Parr, R. Inference in hybrid networks: Theoretical limits and practical algorithms. In *The 17th Conference on Uncertainty in Artificial Intelligence*, Seattle, WA, 2001.

Leskovec, J, Krause, A, Guestrin, C, Faloutsos, C, VanBriesen, J, and Glance, N. Cost-effective outbreak detection in networks. In *13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

Nemhauser, G and Wolsey, L. *Studies on Graphs and Discrete Programming*, chapter Maximizing Submodular Set Functions: Formulations and Analysis of Algorithms, pages 279–301. North-Holland, 1981.

Nemhauser, G, Wolsey, L, and Fisher, M. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

Ostfeld, A and Salomons, E. Optimal layout of early warning detection stations for water distribution systems security. *J. Water Resources Planning and Management*, 130(5):377–385, 2004.

Ostfeld, A and Salomons, E. Sensor network design proposal for the battle of the water sensor networks (bwsn). In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006.

Ostfeld, A, Uber, J. G, Salomons, E, Berry, J. W, Hart, W. E, Phillips, C. A, Watson, J.-P, Dorini, G, Jonkergouw, P, Kapelan, Z, Pierro, F di, Khu, S.-T, Savic, D, Eliades, D, Polycarpou, M, Ghimire, S. R, Barkdoll, B. D, Gueli, R, Huang, J. J, McBean, E. A, James, W, Krause, A, Leskovec, J, Isovitsch, S, Jianhua, Guestrin, C, VanBriesen, J, Small, M, Fischbeck, P, Preis, A, Propato, M, Piller, O, Trachtman, G. B, Wu, Z. Y, and Walski, T. The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms. *submitted to Journal of Water Resources Planning and Management*, 2008.

Preis, A and Ostfeld, A. Multiobjective sensor design for water distribution systems security. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006.

Propato, M and Piller, O. Battle of the water sensor networks. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006.

Rossman, L. A. The epanet programmer's toolkit for analysis of water distribution systems. In *Annual Water Resources Planning and Management Conference*, 1999.

Sviridenko, M. A note on maximizing a submodular set function subject to knapsack constraint. *Operations Research Letters*, 32:41–43, 2004.

Uber, J, Janke, R, Murray, R, and Meyer, P. Greedy heuristic methods for locating water quality sensors in distribution systems. In *Proc. World Water & Environmental Resources Congress, EWRI-ASCE*, Reston, Virginia, 2004.

Watson, J.-P, Greenberg, H. J, and Hart, W. E. A multiple-objective analysis of sensor placement optimization in water networks. In *World Water and Environmental Resources Conference*, Reston VA, 2004.

Wu, Z. Y and Walski, T. Multi objective optimization of sensor placement in water distribution systems. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006.

Proofs

Proof of Theorem 1. Our proof is similar to the analysis of Nemhauser *et al.* (1978). Fix scenario i. We first show that the function $R_i(\mathcal{A}) = \pi_i(\infty) - \pi_i(T(\mathcal{A}, i))$ is submodular. Consider $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{S}$. Let $s \in \mathcal{S} \setminus \mathcal{B}$. We have three cases. (i) $T(s,i) \geq T(\mathcal{A},i)$. Then $T(\mathcal{A} \cup \{s\}) = T(A)$ and $T(\mathcal{B} \cup \{s\}) = T(\mathcal{B})$ and hence $R_i(\mathcal{A} \cup \{s\}) - R_i(\mathcal{A}) = 0 = R_i(\mathcal{B} \cup \{s\}) - R_i(\mathcal{B})$. (ii) $T(\mathcal{B},i) \leq T(s,i) < T(\mathcal{A},i)$. In this case, $R_i(\mathcal{A} \cup \{s\}) - R_i(\mathcal{A}) \geq 0 = R_i(\mathcal{B} \cup \{s\}) - R_i(\mathcal{B})$. Finally, (iii), $T(s,i) < T(\mathcal{B},i)$. In this case, $R_i(\mathcal{A} \cup \{s\}) - R_i(\mathcal{A}) = [\pi_i(\infty) - \pi_i(T(s,i))] - R_i(\mathcal{A}) \geq [\pi_i(\infty) - \pi_i(T(s,i))] - R_i(\mathcal{B}) = R_i(\mathcal{B} \cup \{s\}) - R_i(\mathcal{B})$, where the inequality is due to the nondecreasingness of $R_i(\cdot)$. Hence, for each scenario i, the function R_i is submodular. Now, $R(\mathcal{A}) = \sum_i P(i) R_i(\mathcal{A})$ is a nonnegative linear combination of submodular functions, and hence submodular too. □

Proof of Theorem 2. By reduction from set cover. Let a set $\mathcal{S}$ be given, along with a collection of subsets $\mathcal{B}_1, \ldots, \mathcal{B}_m \subseteq \mathcal{S}$. It is NP-hard to decide whether, for a constant k, there exists a collection of subsets $\mathcal{B}_{i_1}, \ldots, \mathcal{B}_{i_k}$ such that their union covers $\mathcal{S}$ (Garey and Johnson, 2003). We can turn such an instance into a sensor placement problem, where we have a contamination scenario for each element $i \in \mathcal{S}$, and a sensor corresponding to each subset $\mathcal{B}_i$. For a constant k, the penalty minimization problem is the problem of selecting a set of sensors which minimize the number of undetected scenarios. Hence, there exists a set cover of size k if and only if there exists a sensor placement with expected penalty of 0. If we had a sensor placement algorithm which, for a constant k, would be guaranteed to obtain a solution whose expected penalty is some multiple $\alpha(n)$ of the minimum penalty, where $\alpha(n)$ only depends on the problem size n, then we could use this algorithm to decide

the set cover problem, implying that $P = NP$. $\square$

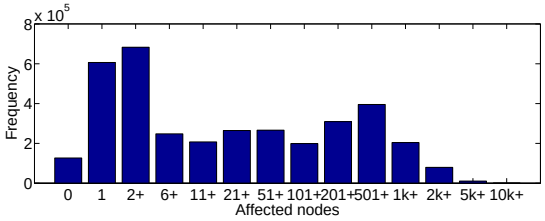
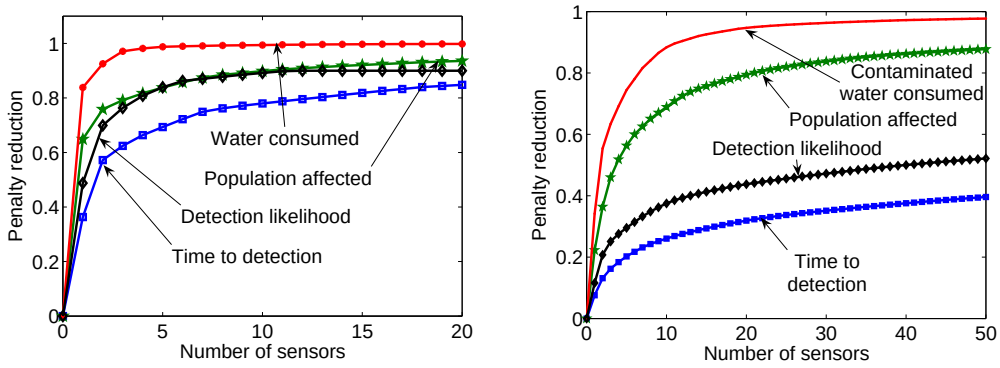
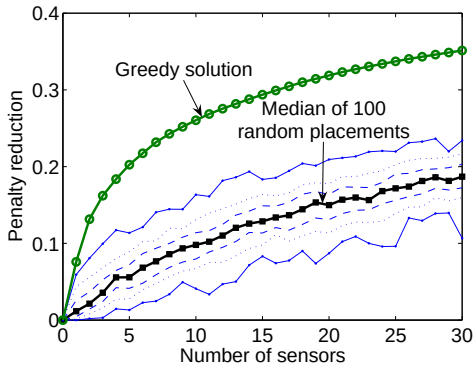


Figure 1: Number of scenarios affecting a given number of nodes.

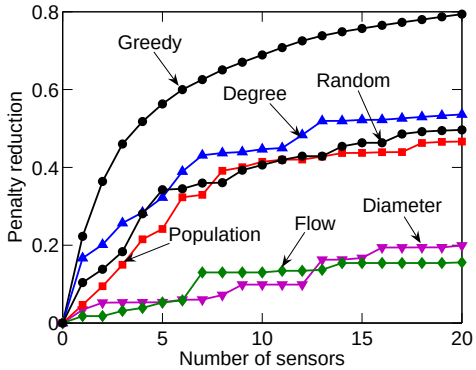


(a) BWSN1

(b) BWSN2



(c) Z_1



(d) Z_2

Figure 2: Four different penalty reduction functions optimized for the small (a) and large (b) network. The diminishing returns property results in a concave performance curve. Comparison of optimized and random placements (100 random trials). (c) Minimizing detection time (Z_1), (d) minimizing the contaminated water consumed (Z_3).

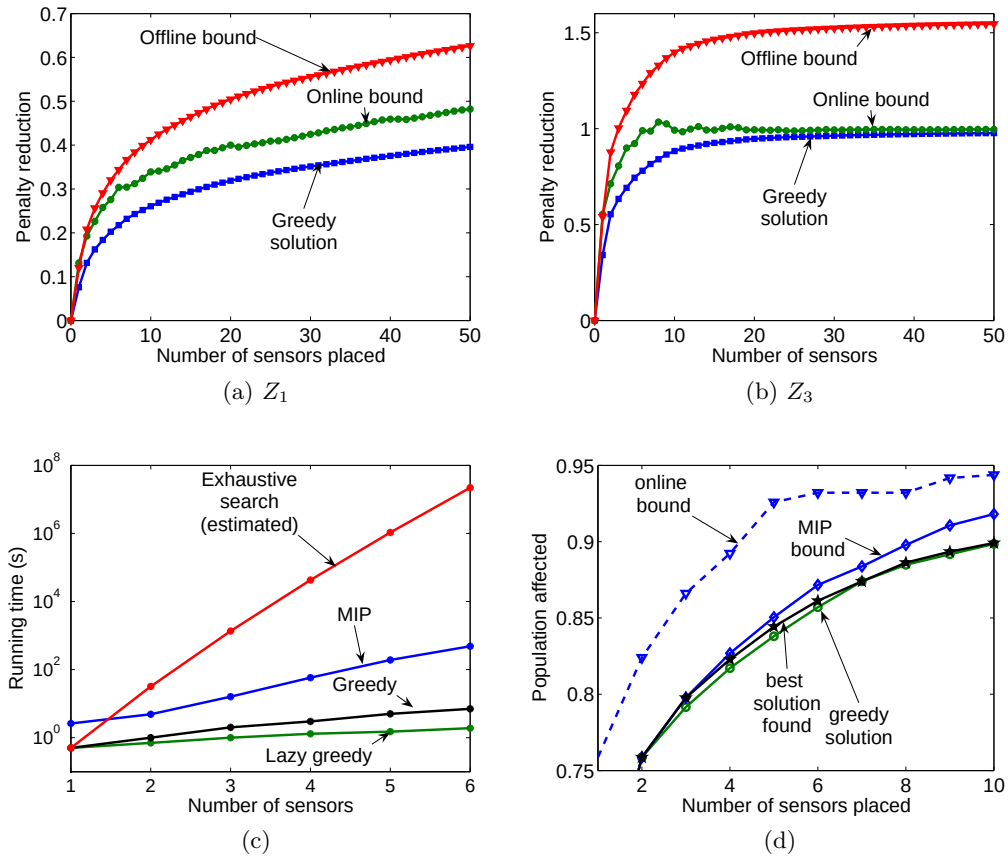
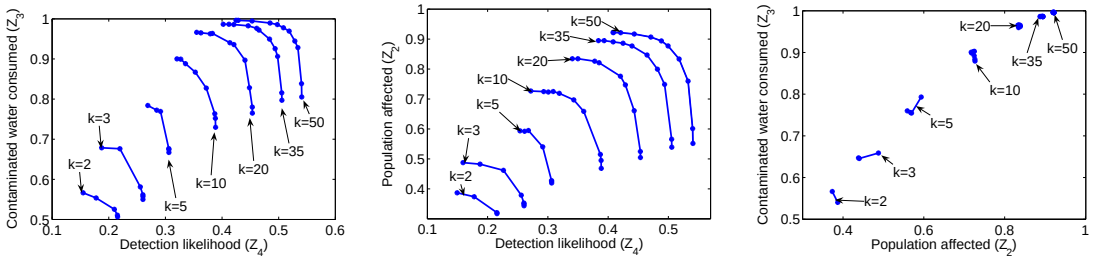
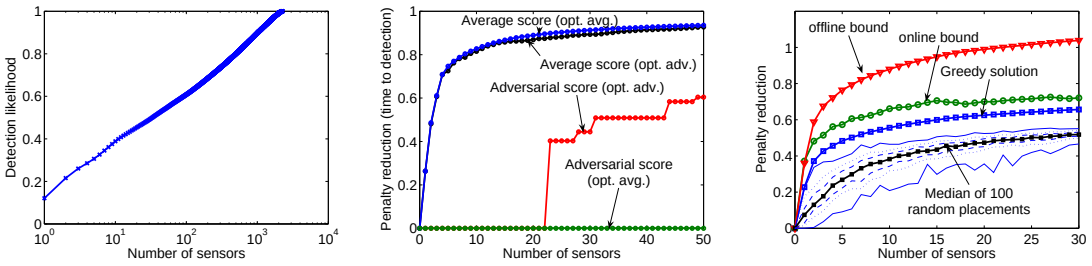


Figure 3: Greedy scores (higher is better) and bounds, for BWSN2, Z_1 (a) and Z_3 (b). (c) comparison of running times for different algorithms (notice the log-scale of the plot) for optimizing Z_1 on BWSN1. (d) online bounds obtained for Z_2, in BWSN1, using MIP are tighter than online bounds due to submodularity.



(a) (b) (c)

Figure 4: Multicriteria analyses, trading off pairs of objective functions on (BWSN2). Higher values on both axes are better.



(a) (b) (c)

Figure 5: (a) For the large network (BWSN2), we need an exponentially increasing number of sensors to eventually detect all intrusions. (a) Results on optimizing adversarial objectives on the large network (BWSN2). (c) Greedy scores, bounds and 10, 25, 50, 75, 90 percentiles of 100 random placements when optimizing the sum of equally weighted Z scores on NW3.

Network	BWSN1	BWSN2	NW3
# Scenarios	37,152	3,602,776	120,000
Memory use	14.7 MB	16.3 GB	2.0 GB
Run time (min)	0.05	41.2	2.73

Table 1: Memory use and running time for placing 10 sensors using greedy algorithm with lazy evaluations.

List of Figures