SPOT: A Sensor Placement Optimization Toolkit for Drinking Water Contaminant Warning System Design

William E. Hart[1], Jonathan W. Berry[2], Regan Murray[3],
Cynthia A. Phillips[4], Lee Ann Riesen[5], Jean-Paul Watson[6]

Abstract

We present SPOT, a Sensor Placement Optimization Tool. SPOT provides a toolkit that facilitates research in sensor placement optimization and enables the practical application of sensor placement solvers to real-world CWS design applications. This paper provides an overview of SPOT's key features, and then illustrates how this tool can be flexibly applied to solve a variety of different types of sensor placement problems.

1 Introduction

Contamination warning systems (CWS) have been proposed as a promising approach for the early detection of contamination events in drinking water distribution systems [1, 2, 3, 22]. The overall goal of a CWS is to detect contamination events in time to reduce potential public health and economic consequences. Online sensors will be a critical component of a CWS, and a key element of CWS design is the optimization of the number and locations of sensors at utility.

We describe SPOT, a Sensor Placement Optimization Tool for CWS design in water distribution systems. SPOT integrates a variety of solvers for sensor placement that have been developed by Sandia National Laboratories and the Environmental Protection Agency, along with many academic collaborators [5, 6, 7, 8, 9, 10, 11, 12, 18, 23, 24]. SPOT includes (1) general-purpose heuristic solvers that consistently locate optimal solutions in minutes, (2) integer-programming heuristics that find solutions of provable quality, (3) exact solvers that find globally optimal solutions, and (4) bounding techniques that can evaluate solution optimality. SPOT uses a generic solver formulation that allows a user to specify a wide range of performance objectives for contaminant warning system design, including population-based public health measures, time to detection, extent of contamination, volume consumed and number of failed detections. SPOT has been integrated into the Environmental Protection Agency's TEVA-SPOT tool, which supports graphical analysis of threat assessment and sensor placement. TEVA-SPOT has been used to develop sensor placement designs for several large U.S. cities.

In general, a user may care about more than one of these performance objectives, so SPOT allows for the specification of constraints that ensure that multiple performance objectives are simultaneously satisfied. SPOT facilitates the interactive design and analysis of sensor placement designs. For example, a SPOT user can integrate expert knowledge during the design process by

[1] Sandia National Laboratories, PO Box 5800, Albuquerque, NM 87185-1318; PH: 505-844-2217; FAX: 505-845-7442; Email: wehart@sandia.gov
[2] Sandia National Laboratories, PO Box 5800, Albuquerque, NM 87185-1318; jberry@sandia.gov
[3] USEPA, 26 W Martin Luther King Dr, Cincinnati OH 45268; Murray.Regan@epa.gov
[4] Sandia National Laboratories, PO Box 5800, Albuquerque, NM 87185-1318; caphill@sandia.gov
[5] Sandia National Laboratories, PO Box 5800, Albuquerque, NM 87185-1318; lafisk@sandia.gov
[6] Sandia National Laboratories, PO Box 5800, Albuquerque, NM 87185-1318; jwatson@sandia.gov

specifying required sensor placements or designating network locations as forbidden. Further, cost considerations can be integrated by limiting the design with user-specified installation costs at each location.

This paper summarizes the key features of SPOT, and describes the canonical problem formulation solved by SPOT solvers. We illustrate the flexibility of this framework in several simple examples. These examples illustrate how SPOT is used to optimize performance objectives, minimize costs, perform multi-objective analysis, constrain sensor placements, and perform optimization with limited memory resources.

2 SPOT Overview

SPOT provides a toolkit that facilitates research in sensor placement optimization and enables the practical application of sensor placement solvers to real-world CWS design applications. A wide range of sensor placement optimization formulations and solver techniques have been developed for CWS design in drinking water systems [6, 10, 13, 14, 15, 16, 17, 20, 21]. Although SPOT certainly leverages much of this work, the goal of SPOT is to develop a toolkit that can incorporate and evaluate a wide range of sensor placement solvers on a broad class of CWS design problems. The following sections provide an overview of SPOT that illustrates the flexibility of this toolkit.

2.1 Contaminants and Contaminant Impacts

SPOT supports a very generic problem formulation that relies on contaminant impacts derived from an *external* contaminant transport calculation and impact assessment. An impact assessment is assumed to come from a set of contamination events, each of which represents a single location and time of contamination. For example, EPA's TEVA tool calculates contaminant and public health impacts using EPANET for hydraulic and water quality calculations and epidemiological models to estimate exposure and disease progression [19].

2.2 Performance Objectives

There are many competing CWS design objectives, such as minimizing exposure to contaminants, minimizing illness, minimizing the spatial extent of contamination, minimizing sensor detection time, and minimizing CWS costs. SPOT treats each of these objectives as a separate contaminant impact assessment (e.g. you can have a different impact assessment for the extent of contamination and the time to detection for a single contaminant). One or more contaminant impact assessments can be integrated into SPOT, thereby allowing for trade-offs between different contaminants, as well as different impacts for single contaminants.

2.3 Problem Formulations

The canonical SPOT formulation is to minimize the mean impact of contamination for CWS design. SPOT also allows for user-specified weights on the expected impact. Further, SPOT solvers can optimize other performance objectives like worst-case impact and related robustness performance measures that trade-off mean- and worst-case impacts.

2.4 Limited-Memory Requirement

The expected application platform for SPOT is 32-bit MS Windows workstations. Consequently, SPOT supports sensor placement solvers that can work with 4GB of memory, even for water distribution networks with 10,000s of pipes and junctions. Although SPOT's integer programming solvers require more than 4GB of memory for these large networks, integer programming heuristics have been developed that solve a related, reduced-memory formulation.

2.5 Flexible Solvers

The rapid solution of large sensor placement optimization is essential to provide users the ability to explore a wide range of performance objectives and design trade-offs. Fast heuristic methods are included in SPOT that can reliably and quickly solve the mean impact sensor placement formulation. Rigorous performance guarantees are also needed to ensure that the final solution generated by SPOT is the best one possible. In cases where this cannot be guaranteed, methods have been developed that can quantify the optimality of the final solution.

3 A Canonical Sensor Placement Formulation

A key principle in the development of SPOT has been the development of mathematical sensor placement formulations that guide the development of subsequent solvers. The canonical formulation used in SPOT is the mean-impact formulation, which can be expressed as the following integer program:

$$
\begin{array}{lll}
\min & \sum_{a \in \mathcal{A}} \alpha_a \sum_{i \in \mathcal{L}_a} d_{ai} x_{ai} & \\
\text{s.t.} & \sum_{i \in \mathcal{L}_a} x_{ai} = 1 & \forall a \in \mathcal{A} \\
 & x_{ai} \leq s_i & \forall a \in \mathcal{A}, i \in \mathcal{L}_a \\
 & \sum_{i \in L} s_i \leq p & \\
 & s_i \in \{0, 1\} & \forall i \in L \\
 & 0 \leq x_{ai} \leq 1 & \forall a \in \mathcal{A}, i \in \mathcal{L}_a
\end{array}
$$

This formulation models the placement of p sensors on a set L vertices, with the objective of minimizing the expected impact of a set $\mathcal{A}$ of contamination events. The binary decision variable s_i for each potential sensor location $i \in L$ equals 1 if a sensor is placed at location i and 0 otherwise. Each contamination event $a \in \mathcal{A}$ has a likelihood α_a such that $\sum_{a \in A} \alpha_a = 1$. Let $\mathcal{L}_a$ be the subset of locations that could possibly be contaminated by event a. For all locations $i \in \mathcal{L}_a$, the impact of the event a is $d_{ai} x_{ai}$, where d_{ai} is a the precomputed contamination impact for this event, when detected at location i, and x_{ai} indicates whether the event has been detected at location i. Berry et al. [10] provide further discussion of this model, including details needed to make the solution of this integer program tractable.

The use of precomputed impact values, d_{ai}, enables the application of this sensor placement formulation to a wide range of performance objectives, since different objectives simply translate into different impact values (e.g. see Watson et al. [23]). Further, this formulation can be used to optimize over the impacts of multiple contaminants. Computation of the mean impact across multiple contaminants reduces to the computation of a single set of impact values. Similarly, the impact of detection delays can be captured in these impact values, and thus this formulation can account for a sophisticated response to sensor detections.

4 Using SPOT

Conceptually, the process of sensor placement can be decomposed into four steps:

1. Performing water quality simulations
2. Computation of impact values
3. Sensor placement optimization
4. Analysis of final sensor placement(s)

Although SPOT contains components supporting all of these steps, our examples illustrate how SPOT can be used for sensor placement optimization (step 3).

The common interface to SPOT's sensor placement solvers is the `sp` script. The following sections illustrate the use of `sp`, to solve the canonical sensor placement formulation described in Section 3 and related sensor placement problems. These examples use EPANET network example 3. The sp script has many different options, but the following are commonly used:

```
--network=<network>
        The name of the network that will be analyzed
--objective=<goal>_<statistic>
        The sensor placement objectives.
--ub=<objective>,<ub-value>
        A constraint on the maximal value of an objective.
--costs=<filename>
        A file containing costs for the installation of sensors
--sensor-locations=<filename>
        A file containing information about whether network ids are feasible
        for sensor placement, and whether a sensor placement is fixed at a
        given location.
--solver=<type>
        Specifies the type of solver that is used to find sensor placement(s).
--compute-bound
        Only compute a bound on the value of the optimal solution.
```

4.1 Example 1 – A Simple Example

Our first example shows how to use a heuristic solver to minimize the extent of contamination (EC) while limiting the number of sensors (NS) to no more than 5.

```
$ ../../../bin/sp --path=../../../bin --path=../mod --network=test1 --
objective=ec --ub=ns,5

Number of Contamination Events:  97
Number of Contamination Impacts: 9458

Validating input for heuristic solver
Number of sensors=5
Objective=ec
Statistic=mean
Impact file=/home/wehart/src/spot-th/builds/teva-spot-
s/spot/packages/sp/test/test1_ec.impact
Delay=0
```

```
Running iterated descent heuristic for *perfect* sensor model
Iterated descent completed - sensors written to file=test1.sensors
-------------------------------------------------------------------
Sensor placement id:        846930886
Number of sensors:          5
Total cost:                 0
Sensor node IDs:            19 28 54 63 75
Sensor junctions:           119 141 193 207 239

Impact File:                /home/wehart/src/spot-th/builds/teva-spot-
s/spot/packages/sp/test/test1_ec.impact
Number of events:           236
Min impact:                 0.0000
Mean impact:                8478.9674
Lower quartile impact:      0.0000
Median impact:              6949.0000
Upper quartile impact:      12530.0000
Value at Risk (VaR) (  5%): 25960.0000
TCE                 (  5%): 33323.2833
Max impact:                 42994.8000
-------------------------------------------------------------------
```

The first output lines summarize the statistics of this network example. Subsequent lines until the first dashed line are generated by the heuristic optimization solver; for simplicity, this text will be omitted in subsequent examples. The information between the dashed lines is a summary of the best sensor placement found. The sensor junctions correspond to the junction labels provided by the user (e.g. from the EPANET input file). Various impact statistics are given, including the mean (which is optimized here), and maximum (worst) impact amongst all contamination events. The VaR and TCE statistics are robust optimization criteria. VaR is the value of the 95 quartile (the impact that is greater than 95% of the impacts), and TCE is the expectation of the worst 5% of the impacts.

The heuristic solves the sensor placement formulation described in Section 3. Consequently, we evaluate the optimality of this solution by solving the linear programming relaxation of this integer program. This can be done in SPOT using the PICO solver:

```
$ ../../../bin/sp --path=../../../bin --path=../mod --network=test1 --
objective=ec --ub=ns,5 --solver=pico --compute-bound

Number of Contamination Events:  97
Number of Contamination Impacts: 9458

Objective lower bound: 8478.96737288
```

This computation demonstrates that the heuristic has found an optimal solution. This can be confirmed by running SP with the PICO solver but without the `compute-bound` option. Without this option, PICO find a globally optimal solution, though for large applications it may take too long to run.

4.2 Example 2 – Optimizing Costs

The canonical SPOT formulation makes the tacit assumption that the cost of each sensor placement is equivalent. In practice this is not likely to be true for a variety of reasons (e.g. see Berry [8]). SPOT supports the generic specification of sensor placement costs using the `costs` option, which is used in conjunction with the 'cost' performance objective. For example, suppose that the file `cost_info` contains the following:

```
119 2.0
141 2.0
193 2.0
207 2.0
239 2.0
__default__ 1.0
```

This cost information can be used to constrain our previous example. In this case, we show that if the original points were too expensive, then we would not use any of them if other sensor placement locations were cheaper.

```
$ ../../../bin/sp --path=../../../bin --path=../mod --network=test1 --
objective=ec --ub=cost,10 --solver=pico --costs=cost_info

Number of Contamination Events:  97
Number of Contamination Impacts: 9458

-------------------------------------------------------------------
Sensor placement id:        13195
Number of sensors:          10
Total cost:                 10
Sensor node IDs:            74 62 38 50 32 16 18 10 24 21
Sensor junctions:           237 206 163 185 149 113 117 101 127 121

Impact File:                /home/wehart/src/spot-th/builds/teva-spot-
s/spot/packages/sp/test/test1_ec.impact
Number of events:           236
Min impact:                 0.0000
Mean impact:                5207.0453
Lower quartile impact:      0.0000
Median impact:              3610.0000
Upper quartile impact:      10416.0000
Value at Risk (VaR) (  5%): 14985.0000
TCE                 (  5%): 17332.3583
Max impact:                 20170.0000
-------------------------------------------------------------------
```

Similarly, SPOT can minimize costs or the number of sensors while maintaining a given performance. The following example minimizes the number of sensors with EC bounded to be no more than 6000.0.

```
$ ../../../bin/sp --path=../../../bin --path=../mod --network=test1 --
objective=ns --ub=ec,5000 --solver=pico

Number of Contamination Events:  97
Number of Contamination Impacts: 9458

-------------------------------------------------------------------
Sensor placement id:        13459
Number of sensors:          10
Total cost:                 0
Sensor node IDs:            75 63 28 38 50 33 54 17 11 21
Sensor junctions:           239 207 141 163 185 151 193 115 103 121

Impact File:                /home/wehart/src/spot-th/builds/teva-spot-
s/spot/packages/sp/test/test1_ec.impact
Number of events:           236
Min impact:                 0.0000
Mean impact:                4675.6708
Lower quartile impact:      0.0000
Median impact:              3610.0000
Upper quartile impact:      6984.0000
Value at Risk (VaR) (  5%): 14490.0000
TCE                 (  5%): 17874.4083
Max impact:                 22450.0000
-------------------------------------------------------------------
```

4.3 Example 3 – Multi-Objective Analysis

The canonical SPOT formulation optimizes a single objective subject to a cost constraint. This formulation can be generalized to include additional constraints that limit the impact of other performance objectives. The following example illustrates how the first example can be extended to re-optimize the solution with respect to a different objective. First, we evaluate the final sensor placement generated by the heuristic solver using the mass consumed (MC) impact values.

```
$ ../../../bin/evalsensor test1.sensors test1_mc.impact
-------------------------------------------------------------------
Sensor placement id:        846930886
Number of sensors:          5
Total cost:                 0
Sensor node IDs:            19 28 54 63 75
Sensor junctions:

Impact File:                test1_mc.impact
Number of events:           236
Min impact:                 0.0000
Mean impact:                43636.7076
Lower quartile impact:      220.0020
Median impact:              1909.9500
Upper quartile impact:      105031.0000
Value at Risk (VaR) (  5%): 144271.0000
TCE                 (  5%): 144345.0000
Max impact:                 144477.0000
-------------------------------------------------------------------
```

Since this sensor placement was generated by optimizing for EC, the MC impacts are likely to be non-optimal. The following optimization minimizes MC while keeping EC close to its optimal value. Note that statistics for both impact values are reported, which show that we have satisfied this constraint while improving (minimizing) the mean MC impact.

```
$ ../../../bin/sp --path=../../../bin --path=../mod --network=test1 --
objective=mc --ub=ns,5 --ub=ec,9000.0

Number of Contamination Events:  97
Number of Contamination Impacts: 9458

-------------------------------------------------------------------
Sensor placement id:        1681692777
Number of sensors:          5
Total cost:                 0
Sensor node IDs:            2 19 54 63 75
Sensor junctions:           15 119 193 207 239

Impact File:                /home/wehart/src/spot-th/builds/teva-spot-
s/spot/packages/sp/test/test1_mc.impact
Number of events:           236
Min impact:                 0.0000
Mean impact:                39963.3542
Lower quartile impact:      220.2160
Median impact:              1546.2700
Upper quartile impact:      86439.9000
Value at Risk (VaR) (  5%): 144271.0000
TCE                 (  5%): 144276.3333
Max impact:                 144335.0000

Impact File:                /home/wehart/src/spot-th/builds/teva-spot-
s/spot/packages/sp/test/test1_ec.impact
Number of events:           236
Min impact:                 0.0000
Mean impact:                8844.6453
Lower quartile impact:      0.0000
Median impact:              6949.0000
Upper quartile impact:      13502.0000
Value at Risk (VaR) (  5%): 25960.0000
TCE                 (  5%): 35007.4500
Max impact:                 42994.8000
-------------------------------------------------------------------
```

SPOT supports an arbitrary number of such constraints. However, in practice only one additional side constraint can be effectively solved.

4.4 Example 4 – Constraining Sensor Locations

Properties of the sensor locations can be specified with the `sensor-locations` option. This option specifies a file that can control whether sensor locations are feasible or infeasible, and fixed or unfixed. For example, suppose that the file `locations` contains

```
infeasible 119 141 193 207 239
fixed 161
```

Then the following example shows that these restrictions lead to a solution with a value that is worse than the first example (above) which has an optimal value of `8478.9674`.

```
$ ../../../bin/sp --path=../../../bin --path=../mod --network=test1 --
objective=ec --ub=ns,5 --sensor-locations=locations

Number of Contamination Events:  97
Number of Contamination Impacts: 9458

-----------------------------------------------------------------
Sensor placement id:        1804289383
Number of sensors:          5
Total cost:                 0
Sensor node IDs:            17 33 37 50 66
Sensor junctions:           115 151 161 185 211

Impact File:                /home/wehart/src/spot-th/builds/teva-spot-
s/spot/packages/sp/test/test1_ec.impact
Number of events:           236
Min impact:                 0.0000
Mean impact:                9338.7119
Lower quartile impact:      0.0000
Median impact:              7640.0000
Upper quartile impact:      14120.0000
Value at Risk (VaR) (  5%): 27335.0000
TCE                 (  5%): 32282.3000
Max impact:                 45300.0000
-----------------------------------------------------------------
```

4.5 Example 5 - Limited-Memory Solvers

A key issue for the practical application of SPOT is the memory requirements for SPOT's heuristic and integer programming solvers. The memory required for sensor placement solvers can be significantly impacted by the number of junctions at which sensor placement is feasible, the number of contamination events that are modeled, and the extent to which these events impact a large fraction of the network. SPOT's solvers can be effectively applied to applications with 100s of junctions and pipes on 32-bit workstations with 4GB of RAM. However, larger applications often exceed available memory resources. SPOT includes several mechanisms that reduce the solver memory requirements, at the expense of a slower run time and possible inaccuracies in the problem formulation. These modified solvers are able to optimize applications with 10,000s of junctions and pipes using only a few gigabytes of memory.

Two related strategies have been developed for applying SPOT's integer programming solver with limited memory; see Berry et al. [4] for further details on these strategies. The first strategy involves combining contamination events that have very similar impacts. This approach integrates two or more impact files into an aggregated impact file that contains representative impact events. Thus, this can be viewed as a preprocessing step for SPOT's optimizers.

The second strategy involves a reformulation of the integer programming formulation to reduce the number of decision variables and constraints that need to be processed. This method leverages the fact that many impact values are similar, and thus is make sense to aggregate

similar impact values together. Two methods are supported for this aggregation process in SPOT: (ratio) – limits the ratio of the smallest impact to largest impact in a set that is aggregated together, and (percent) – uses an aggregation threshold that is based on the percent of the difference of the largest to smallest possible impacts.

Table 1 illustrates the uses of using these two aggregation strategies on the size of the integer program, the total memory needed to solve the problem (using the PICO integer programming solver included with SPOT). Further, this table shows that the final solution computed with the ratio aggregation can be used to compute a lower bound on the value of the best sensor placement. Note that the zero percent aggregation is the default mechanism used in SPOT, since this performs a simple aggregation without impacting the final solution.

Percent	Ratio	# Variables	# Constraints	# Nonzeros	Final Solution	Lower Bound	Memory (Kb)
None	None	9558	9466	37421	21781.98	21781.98	39704
0	0.00	4505	4413	22184	21781.98	21781.98	25904
25	0.00	700	608	9127	21804.45	0	18568
50	0.00	571	479	8255	21887.27	0	18028
75	0.00	562	470	7102	21781.98	0	18164
100	1.00	4505	4413	22184	21781.98	21781.98	25904
100	0.75	2010	1918	13089	21781.98	15922.66	20164
100	0.50	1452	1360	11102	21781.98	10513.44	19496
100	0.25	1084	992	9326	21781.98	5125.84	19096
100	0.12	946	854	8568	21804.45	2429.90	18964

Table 1. Impact of percent and ratio aggregation on a small sensor placement example (EPANET network 3). 100-percent aggregation with 0-ratio results are omitted, since the integer program becomes trivially non-interesting in that case.

The sensor placement heuristic solvers generally require much less memory than the integer programming solver. However, these solvers employ a large matrix of precomputed values that speeds up the solution of large problems. The `heuristic-representation` option in SPOT enables this precomputation to be stored as a sparse matrix, which trades off runtime performance for space savings.

5 Discussion

These examples illustrate the flexibility of SPOT's solvers, and the broad applicability of SPOT's canonical problem formulation. SPOT also supports advanced model formulations for robust optimization and modeling sensor placement with imperfect sensor detections. However, these capabilities cannot be effectively applied to large-scale real-world applications, and thus they remain a subject of ongoing research.

SPOT sensor placement solvers have been integrated into the TEVA-SPOT tool. This tool has been used for CWS design studies at several large U.S. cities, and it is being used in ongoing studies within the EPA's Water Sentinel program. We expect that much of this software will be released under an open-source license in 2007. Contact William Hart (wehart@sandia.gov) for further information.

Acknowledgements

This work was funded by the National Homeland Security Research Center at the United States Environmental Protection Agency. We thank Rob Janke and Tom Taxon for feedback on SPOT and for their patience while issues are being resolved. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Bibliography

[1] *Interim Voluntary Guidelines for Designing an Online Contaminant Monitoring System*, American Society of Civil Engineers, 2004.

[2] *Review of State-of-the-Art Early Warning Systems for Drinking Water*, U. S. Environmental Protection Agency, 2005.

[3] *WaterSentinel System Architecture*, U. S. Environmental Protection Agency, 2005.

[4] J. Berry, R. Carr, W. E. Hart and C. A. Phillips, *Scalable water network sensor placement via aggregation*, *World Environment and Water Resources Congress*, Tampa Florida, 2007.

[5] J. Berry, R. D. Carr, W. E. Hart, V. J. Leung, C. A. Phillips and J.-P. Watson, *On the placement of imperfect sensors in municipal water networks*, *Proc. Water Distribution System Symposium*, 2006.

[6] J. Berry, L. Fleischer, W. E. Hart, C. A. Phillips and J.-P. Watson, *Sensor Placement in Municipal Water Networks*, J. Water Planning and Resources Management, 131 (2005), pp. 237-243.

[7] J. Berry, W. E. Hart, C. A. Phillips and J. Uber, *A general integer-programming-based framework for sensor placement in municipal water networks*, *Proc. World Water and Environment Resources Conference*, 2004.

[8] J. Berry, W. E. Hart, C. A. Phillips, J. G. Uber and T. M. Walski, *Water quality sensor placement in water networks with budget constraints*, *Proc. World Water and Environment Resources Conference*, 2005.

[9] J. Berry, W. E. Hart, C. A. Phillips and J.-P. Watson, *Validation and assessment of integer programming placement models*, *Proc. World Water and Environment Resources Conference*, 2005.

[10] J. Berry, W. E. Hart, C. E. Phillips, J. G. Uber and J.-P. Watson, *Sensor placement in municiple water networks with temporal integer programming models*, J. Water Resources Planning and Management, 132 (2006), pp. 218-224.

[11] J. W. Berry, W. E. Hart and C. A. Phillips, *Scalability of integer programming computations for sensor placement in municipal water networks*, *Proc. World Water and Environmental Resources Congress*, American Society of Civil Engineers, 2005.

[12] R. Carr, H. J. Greenberg, W. E. Hart, G. Konjevod, E. Lauer, H. Lin, T. Morrison and C. A. Phillips, *Robust optimization of contaminant sensor placement for community water systems*, Mathematical Programming Series B, 107 (2006), pp. 337-356.

[13] R. A. Deininger, *The Survival of Father Rhine*, Journal of the American Water Works Association, 70 (1987), pp. 78-84.

[14] A. Kessler, A. Ostfeld and G. Sinai, *Detecting Accidental Contaminations in Municipal Water Networks*, Journal of Water Resources Planning and Management, 124 (1998), pp. 192-198.

[15] A. Kumar, M. L. Kansal and G. Arora, *Discussion of `Detecting Accidental Contaminations in Municipal Water Networks'*, Journal of Water Resources Planning and Management, 125 (1999), pp. 308-310.

[16] B. H. Lee and R. A. Deininger, *Optimal Locations of Monitoring Stations in Water Distribution System*, Journal of Environmental Engineering, 118 (1992), pp. 4-16.

[17] B. H. Lee, R. A. Deininger and R. M. Clark, *Locating Monitoring Stations in Water Distribution Systems*, Journal, Am. Water Works Assoc. (1991), pp. 60-66.

[18] R. Murray, W. Hart and J. Berry, *Sensor network design for contamination warning systems: Tool and applications*, *Proc. AWWA Congress on Water Security*, 2006.

[19] R. Murray, J. G. Uber and R. Janke, *Model for estimating acute health impacts from consumption of contaminated drinking water*, J. Water Planning and Resources Management, 132 (2006), pp. 293-299.

[20] 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 (2004), pp. 377-385.

[21] M. Propato, O. Piller and J. Uber, *A Sensor Location Model to Detect Contaminations in Water Distribution Networks*, *Proc. World Water and Environmental Resources Congress*, American Society of Civil Engineers, 2005.

[22] A. Roberson and K. Morley, *Contamination Warning Systems for Water: An Approach for Providing Actionable Information to Decision-Makers*, AWWA, 2005.

[23] J.-P. Watson, H. J. Greenberg and W. E. Hart, *A multiple-objective analysis of sensor placement optimization in water networks*, *Proc. World Water and Environment Resources Conference*, 2004.

[24] J.-P. Watson, W. E. Hart and R. Murray, *Formulation and optimization of robust sensor placement problems for contaminant warning systems*, *Proc. Water Distribution System Symposium*, 2006.