Mathematical Programming manuscript No.
(will be inserted by the editor)

Robert D. Carr · Harvey J. Greenberg · William E. Hart · Goran Konjevod · Erik Lauer · Henry Lin · Tod Morrison · Cynthia A. Phillips

Robust Optimization of Contaminant Sensor Placement for Community Water Systems

the date of receipt and acceptance should be inserted later

Abstract We present a series of related robust optimization models for placing sensors in municipal water networks to detect contaminants that are ma-

Robert D. Carr
Discrete Algorithms and Math Department
Sandia National Laboratories
E-mail: rdcarr@sandia.gov

Harvey J. Greenberg
Mathematics Department
University of Colorado at Denver
E-mail: Harvey.Greenberg@cudenver.edu

William E. Hart
Discrete Algorithms and Math Department
Sandia National Laboratories
E-mail: wehart@sandia.gov

Goran Konjevod
Computer Science Department
Arizona State University
E-mail: goran@asu.edu

Erik Lauer
Electrical and Computer Engineering Department
University of New Mexico
E-mail: elauer@sandia.gov

Henry Lin
University of California at Berkeley
E-mail: henrylin@eecs.berkeley.edu

Tod Morrison
Mathematics Department
University of Colorado at Denver
E-mail: tmorriso@math.cudenver.edu

Cynthia A. Phillips
Discrete Algorithms and Math Department Sandia National Laboratories
E-mail: caphill@sandia.gov

liciously or accidentally injected. We formulate sensor placement problems as mixed-integer programs, for which the objective coefficients are not known with certainty. We consider a restricted absolute robustness criteria that is motivated by natural restrictions on the uncertain data, and we define three robust optimization models that differ in how the coefficients in the objective vary. Under one set of assumptions there exists a sensor placement that is optimal for all admissible realizations of the coefficients. Under other assumptions, we can apply sorting to solve each worst-case realization efficiently, or we can apply duality to integrate the worst-case outcome and have one integer program. The most difficult case is where the objective parameters are bilinear, and we prove its complexity is NP-hard even under simplifying assumptions. We consider a relaxation that provides an approximation, giving an overall guarantee of near-optimality when used with branch-and-bound search. We present preliminary computational experiments that illustrate the computational complexity of solving these robust formulations on sensor placement applications.

1 Introduction

Combinatorial optimization techniques need to address modeling and data uncertainties in many applications. As early as the 1950s, Dantzig [7] introduced stochastic programming to deal with *aleatory uncertainty*, which describes the inherent variation associated with the system being modeled [10, 16, 17].

More recently, researchers have developed robust optimization methods [12] to deal with *epistemic uncertainty*, which describes our lack of knowledge about information in our model [10, 16, 17]. For example, a common assumption is that the coefficients in the objective function of the problem are uncertain in the sense that they each can assume any value within a finite interval. Interval data occurs often in practice [8, 12], and when quantitative parameters have a subjective nature, interval values can be used interactively to provide more intuition about the model.

Robust optimization methods generally seek a solution that minimizes some measure of worst performance with respect to the uncertainty in the data. Commonly studied criteria for robust optimization are *absolute robustness* (or minimax), and *robust deviation* (or minimax regret). Yaman, Karason and Pinar [23] survey recent research for these methods and observe that many absolute robust formulations of problems with interval uncertainties can be solved with little more difficulty than the deterministic case. Bertsimas and Sim [5, 6] adopt the interval model of uncertainty and consider a restricted version of the absolute robustness criterion. This model of robustness limits the conservativeness of the robust solution by arguing that it is quite unlikely that all data elements will assume their worst possible values simultaneously; both the absolute robustness and robust deviation criteria may find solutions that have this property. Furthermore, solving a 0-1 mixed-integer linear program (MILP) under this model is no more difficult than solving the original problem.

In this paper we consider a version of the absolute robustness criterion that is naturally restricted by properties of the uncertain data. Specifically, we consider the case where the uncertain coefficients sum to a constant value. This restricted absolute robustness criterion is motivated by our recent work with MILP formulations for sensor placement in water distribution networks [4,3,21]. These MILPs rely on information like attack probabilities and water consumption statistics that are difficult to assess in detail, but for which we have good aggregate estimates. For example, water utilities have little information about the water consumption within a given household at a given hour, but they have accurate information about total water consumption within the entire water distribution network.

We analyze three cases of this restricted absolute robustness criterion that are motivated by this water security application:

- **unweighted uncertainty**: the objective has the form $\sum_{ij} \alpha_i x_{ij}$, for uncertain coefficients α_i;
- **linearly weighted uncertainty**: the objective has the form $\sum_{ij} \alpha_i p_j x_{ij}$, for uncertain coefficients α_i (p_j known with certainty); and,
- **bilinearly weighted uncertainty**: the objective has the form $\sum_{ij} \alpha_i \delta_j x_{ij}$, for uncertain coefficients α_i and δ_j.

We argue that the constant-sum constraints on the uncertain parameters make the solution to these problems more realistic than formulations with a simple absolute robustness criterion. Furthermore, these problem formulations do not require a user-defined parameter to restrict the data uncertainties, so the robustness trade-off in this problem is more intrinsic than the trade-offs considered in the restricted absolute robustness models developed by Bertsimas and Sim [5,6].

The rest of this paper is organized as follows. Section 2 describes and motivates the three robust MILP formulations. The subsequent three sections analyze these models and present some preliminary computational experience using them. We conclude with a discussion of avenues for further research. The mathematical programming terms are generally defined here as needed, but one can consult the *Mathematical Programming Glossary* [9].

2 Motivation for Robust MILP Models

Recent terrorist attacks have heightened concerns about whether community water systems are sufficiently well protected to ensure a safe and reliable supply of drinking water in the United States and around the world. Consequently, there is growing interest in the use of contaminant sensors to provide ongoing monitoring of water quality. A good sensor placement maximizes the information available for contamination control and remediation across a wide range of possible contamination scenarios, so that the fewest users consume contaminated water. A variety of MILP formulations have been developed to identify good sensor placement configurations [4,3, 14,21]. Berry et al. [4,3] have recently solved moderately large MILP models of sensor-placement problems.

We model an attack as the release of a large volume of harmful contaminant at a single junction in the network. For any particular attack, we assume that all points downstream of the release point can be contaminated. Let $\mathcal{R}$ be the set of pairs of junctions (i, j) such that junction j is downstream of junction i.[1] The primary decision variables for optimization are where to place each of a given number of sensors. Secondary binary decision variables are x_{ij}, for $(i, j) \in \mathcal{R}$, where $x_{ij} = 1$ if a contaminant injected at junction i can reach junction j without passing a sensor. (As will be evident, these secondary variables are completely determined by the sensor placements, but they appear as decision variables from a computational view.) Let X denote the set of feasible 0-1 x-values.

Consider the following parameter vectors over the nodes: (1) α_i is the probability of an attack at junction i, and (2) δ_i is the number of people who consume water at junction i. Note that α is not a probability in the classical sense and is sometimes called an *attack weight*. It is estimated from expert judgement about the vulnerabilities in the network. We estimate δ from census data. If the node i represents a contracted sub-network, then δ_i is the sum of the estimated population numbers for all nodes in the sub-network. We assume that all people at a contaminated node are potentially exposed.

The following two problems illustrate the data uncertainties that arise in these applications:

1. Minimize the expected extent of network contamination, as defined by the number of pipe junctions that become contaminated

$$\text{NC} : \min_{x \in X} \sum_{(i,j) \in \mathcal{R}} \alpha_i x_{ij},$$

2. Minimize the expected population exposed

$$\text{PE} : \min_{x \in X} \sum_{(i,j) \in \mathcal{R}} \alpha_i \delta_j x_{ij}.$$

In practice, we do not know the values of α and δ. Although water utilities can accurately estimate the total population served by their water distribution network, most utilities do not currently maintain detailed statistics about the fraction of the population that is consuming water at each junction. Similarly, risk assessment methodologies provide a coarse assessment of attack weights.

Let $\hat{\alpha}$ and $\hat{\delta}$ denote specified values of α and δ, respectively. These are sometimes called the *central values* (viz., most likely values, or best estimates). The central MILP is thus: $\min_{x \in X} \sum_{(i,j) \in \mathcal{R}} \hat{\alpha}_i \hat{\delta}_j x_{ij}$. As we let α and δ deviate from their central values, their possible values are limited by *constant-sum constraints*, $\sum_i \alpha_i = \sum_i \hat{\alpha}_i = 1$ and $\sum_j \delta_j = \sum_j \hat{\delta}_j$, respectively.

In the next two sections, we consider robust formulations for an absolute robustness criterion that is restricted in this sense. For objectives like NC,

[1] To simplify our presentation, we assume a stable flow pattern for water in this paper. In particular, the models that we describe do not explicitly account for temporal effects. Berry et al. [4,3] describe more detailed IP models.

we show that the solution to a specific class of restricted absolute robustness problems is exactly the solution to the central MILP. More complex objectives, like PE, contain terms with products of uncertain parameters ($\alpha_i \delta_j$). The PE model is less complex if population values are presumed known, which is what we consider in Section 3.

We use the following notation in the next sections to define the domain of the robust optimization problems:

$$\mathcal{B}(\hat{c}, L, U) = \{c : L \leq c \leq U,\ \textstyle\sum_k c_k = \sum_k \hat{c}_k\},$$

where we suppose $L \leq \hat{c} \leq U$. The set $\mathcal{B}(\hat{c}, L, U)$ defines a multidimensional interval of uncertainty about a central value, $\hat{c}$, where c can be α or δ.

3 Linearly Weighted Uncertainty

Consider the PE problem with known population values at the nodes. The following robust optimization formulation applies our restricted absolute robustness criteria:

$$\min_{x \in X} \max_{\alpha \in \mathcal{B}(\hat{\alpha}, \underline{\alpha}, \overline{\alpha})} \sum_{(i,j) \in \mathcal{R}} \alpha_i \delta_j x_{ij}, \tag{1}$$

where δ is presumed known. Since each uncertain term is weighted by a constant value, we say that this absolute robustness criteria uses *linearly weighted uncertainty*.

We can apply duality to reformulate problem (1) as follows:

$$\begin{aligned} \min \pi + \mu\overline{\alpha} - \lambda\underline{\alpha} \ : & \ x \in X \\ & \lambda, \mu \geq 0 \\ & \pi + \mu_i - \lambda_i = \sum_{j \in J(i)} \delta_j x_{ij} \text{ for all } i, \end{aligned}$$

where $J(i) = \{j : (i,j) \in \mathcal{R}\}$. The dual variable π is associated with the primal constant-sum constraint, and λ, μ are associated with the lower and upper bounds, respectively.

Thus, we can cast a linearly weighted robust optimization as a single MILP, having replaced the max with min. In particular, this is an augmented MILP formulation, which simply includes an extended objective and some additional side-constraints on dual variables from the maximization subproblem.

Alternatively, instead of casting problem (1) as one MILP, we can decompose it and solve the inner maximization problem to obtain $\alpha^*(x)$ for each x in the outer minimization. The inner maximization problem can be solved simply by sorting the coefficients, which requires no more than $O(|\mathcal{R}| \ln |\mathcal{R}|)$ time. This may be computationally more efficient than the integrated formulation.

4 Unweighted Uncertainty

Consider the NC problem with interval uncertainties on the attack probabilities. The following robust optimization formulation applies our restricted absolute robustness criteria:

$$\min_{x \in X} \max_{\alpha \in \mathcal{B}(\hat{\alpha}, \underline{\alpha}, \overline{\alpha})} \sum_{(i,j) \in \mathcal{R}} \alpha_i x_{ij}. \tag{2}$$

We can solve this problem using the methods described in Section 3 (letting $\delta_j = 1$ for all j), but in this section we consider the restricted case where we have intervals of uncertainty that are proportional to the central value vector. Let $\mathcal{P}(\hat{c}, \varepsilon) = \mathcal{B}(\hat{c}, (1-\varepsilon)\hat{c}, (1+\varepsilon)\hat{c})$ for $\varepsilon \in [0, 1)$. Given this restricted notion of interval uncertainty, we prove that the sensor placement decision for the central attack weight values $\hat{\alpha}$ remains optimal for any allowed variation.

Let $Q(\varepsilon)$ denote a generalized robust optimization problem:

$$\min_{x \in X} \max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx,$$

where X is any subset of binary vectors. The following theorem demonstrates that the solution to the central MILP (where $\hat{c}$ is the coefficient of x) is the solution to a robust formulation that allows percentage deviations within a constant proportion of its central value. Consequently, no additional computational effort is needed to generate a robust solution for these problems.

Theorem 1 *Let $\varepsilon \in [0, 1)$. Then, x^* is an optimal solution to $Q(0)$ if, and only if, x^* is an optimal solution to $Q(\varepsilon)$.*

Proof We begin with some notation and general observations. Let $S = \sum_j \hat{c}_j$. Let $\sigma(x) = \{j : x_j \neq 0\}$ (called the "support set" of x). Also, let $\mathbf{1}$ denote the vector of all ones: $(1, 1, \ldots, 1)^\mathsf{T}$. The following identities follow from the definitions of S and σ: $\hat{c}x = \sum_{j \in \sigma(x)} \hat{c}_j = S - \sum_{j \notin \sigma(x)} \hat{c}_j$, and $\hat{c}(\mathbf{1} - x) = S - \hat{c}x = \sum_{j \notin \sigma(x)} \hat{c}_j$. Let $L = (1 - \varepsilon)$ and $U = (1 + \varepsilon)$.

The dual of $\max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx$ is

$$\begin{aligned} &\min \pi S + U\mu\hat{c} - L\lambda\hat{c} : \lambda, \mu \geq 0, \\ &\pi + \mu_j - \lambda_j = x_j \text{ for all } j = 1, \ldots, n. \end{aligned}$$

The dual variable π is associated with the constant-sum constraint, and λ, μ are associated with the lower and upper bound constraints on c, respectively.

Let x^0 be an optimal solution to $Q(0)$ and let x^ε be an optimal solution to $Q(\varepsilon)$. Our proof divides into two cases, depending on whether $\hat{c}x^0$ is greater or less than $\frac{1}{2}S$.

Case 1. $\hat{c}x^0 \geq \frac{1}{2}S$.

Consider the dual solution $\pi = 1$, $\mu = 0$, and $\lambda^\mathsf{T} = \mathbf{1} - x^0$. This is dual-feasible, where $\lambda \geq 0$ because $x^0 \leq \mathbf{1}$. The dual objective value is

$$\pi S + U\mu\hat{c} - L\lambda\hat{c} = S - L\hat{c}(\mathbf{1} - x^0) = S - L(S - \hat{c}x^0) = \varepsilon S + L\hat{c}x^0.$$

Therefore, we have

$$\max_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^0 \le \varepsilon S + L\hat{c}x^0. \tag{3}$$

Now we define $c_j^\varepsilon = L\hat{c}_j$ for $j \notin \sigma(x^\varepsilon)$. Since we assume that $\hat{c}x^0 \ge \frac{1}{2}S$, it follows that $\hat{c}x^\epsilon \ge \frac{1}{2}S$, which implies that $\hat{c}(\mathbf{1} - x^\varepsilon) \le \frac{1}{2}S$. Consequently, we have

$$\begin{aligned} c^\varepsilon x^\varepsilon &= S - \textstyle\sum_{j\notin\sigma(x^\varepsilon)} c_j^\varepsilon \\ &= S - L\textstyle\sum_{j\notin\sigma(x^\varepsilon)} \hat{c}_j \\ &= S - L(S - \hat{c}x^\varepsilon) = \varepsilon S + L\hat{c}x^\varepsilon, \end{aligned}$$

which gives us the bound:

$$\max_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^\varepsilon \ge \varepsilon S + L\hat{c}x^\varepsilon. \tag{4}$$

Using (3) and (4), we then obtain the following chain of inequalities:

$$\max\nolimits_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^\varepsilon \ge \varepsilon S + L\hat{c}x^\varepsilon \ge \varepsilon S + L\hat{c}x^0 \ge \max\nolimits_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^0 \ge \max\nolimits_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^\varepsilon.$$

Thus, equality must hold throughout. This establishes the following two results:

$$\begin{aligned} \max\nolimits_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^0 &= \max\nolimits_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^\varepsilon && \text{(first = last expression)} \\ \hat{c}x^0 &= \hat{c}x^\varepsilon && \text{(second = third expression and } L > 0), \end{aligned}$$

which completes this case.

Case 2. $\hat{c}x^0 \le \frac{1}{2}S$.

The dual objective value of any dual-feasible solution is an upper bound on the primal value, cx^0. Choose $\pi = 0$, $\mu^\mathsf{T} = x^0$, and $\lambda = 0$. This is clearly dual-feasible, and its dual objective value is $U\hat{c}x^0$. Therefore,

$$\max_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^0 \le U\hat{c}x^0. \tag{5}$$

Now consider the value of $\hat{c}x^\varepsilon$. Suppose $\hat{c}x^\varepsilon \le \frac{1}{2}S$. Then define $c_j^\varepsilon = U\hat{c}_j$ for $j \in \sigma(x^\varepsilon)$, and note that $c^\varepsilon \in \mathcal{P}(\hat{c},\varepsilon)$. This is feasible (i.e., $c^\varepsilon \in \mathcal{P}(\hat{c},\varepsilon)$) because $c^\varepsilon x^\varepsilon \le \frac{1}{2}S$. It follows that $c^\varepsilon x^\varepsilon = U\hat{c}x^\varepsilon$, so we have $\max\limits_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^\varepsilon \ge U\hat{c}x^\varepsilon$. On the other hand, suppose if $\hat{c}x^\varepsilon > \frac{1}{2}S$. Then, define $c_j^\varepsilon = L\hat{c}_j$ for $j \notin \sigma(x^\varepsilon)$, and note that $c^\varepsilon \in \mathcal{P}(\hat{c},\varepsilon)$. It follows from our analysis in Case 1 that $\max\limits_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^\varepsilon \ge \varepsilon S + L\hat{c}x^\varepsilon$. Taken together, this gives us the bound:

$$\max_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^\varepsilon \ge \min\left\{U\hat{c}x^\varepsilon,\, \varepsilon S + L\hat{c}x^\varepsilon\right\}. \tag{6}$$

Using Equations (5) and (6), we then obtain the following chain of inequalities:

$$\begin{aligned} \max\nolimits_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^\varepsilon &\ge \min\left\{U\hat{c}x^\varepsilon, \varepsilon S + L\hat{c}x^\varepsilon\right\} \ge \min\left\{U\hat{c}x^0, \varepsilon S + L\hat{c}x^0\right\} \\ &= U\hat{c}x^0 \ge \max\nolimits_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^0 \ge \max\nolimits_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^\varepsilon. \end{aligned}$$

The equality in this chain follows from our assumption that $\hat{c}x^0 \leq \frac{1}{2}S$. We conclude that equality must hold throughout, and $\max_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^0 = \max_{c\in\mathcal{P}(\hat{c},\varepsilon)} cx^\varepsilon$. Furthermore, this shows that $U\hat{c}x^0 = \min\{U\hat{c}x^\varepsilon, \varepsilon S + L\hat{c}x^\varepsilon\}$ (fourth expression = second), so either $U\hat{c}x^0 = U\hat{c}x^\varepsilon$ or $U\hat{c}x^0 = \varepsilon S + L\hat{c}x^\varepsilon$. In the former case, we have immediately that $\hat{c}x^0 = \hat{c}x^\varepsilon$. In the latter case, we have the following chain of inequalities:

$$U\hat{c}x^0 \leq \varepsilon S + L\hat{c}x^0 \leq \varepsilon S + L\hat{c}x^\varepsilon = U\hat{c}x^0,$$

from which it follows that $\hat{c}x^0 = \hat{c}x^\varepsilon$. Consequently, we conclude $\hat{c}x^0 = \hat{c}x^\varepsilon$.

In terms of the sensor placement problem, this result implies that we can solve problem (2) by solving the central MILP,

$$\min_{x\in X} \sum_{(i,j)\in\mathcal{R}} \hat{\alpha}_i x_{ij},$$

because every optimal solution remains optimal for any variation allowance on the attack weights that are bounded by a common proportion of the central value, $\hat{\alpha}$. This is because the α-maximization increases the objective by a ***proportion*** of S, independently of x. This is what is revealed in the proof and highlights the simplicity of the robust model.

While we have let attack weights (α) be the uncertain parameters, we could let it be population (δ) if we assume a uniform distribution on attack location. This is the case when there is no risk analysis, and one fixes $\alpha_i = \frac{1}{n}$ for all $i = 1, \ldots, n$, where n is the number of nodes. In that case we also have the unweighted model, but the meaning of the objective changes to the max-expected population contamination.

Following Yaman et al. [22], this result may be called a *permanent solution*. They found a spanning tree that remains optimal within interval data; we have a sensor placement that remains optimal under fixed-proportionate interval data and a constant-sum constraint. In our case, the fixed proportion is necessary — Theorem 1 is not true if we consider the more general set of uncertainties defined by $\mathcal{B}(\hat{c}, \underline{c}, \overline{c})$.

5 Bilinear Weighted Uncertainty

Consider the PE problem with interval uncertainties on both the attack weights and population. Although the total population remains fixed, in practice we may not have complete knowledge of the population's geographic distribution. Consequently, there may be uncertainties in the values of the δ_i. The following robust optimization formulation applies our restricted absolute robustness criteria considering uncertainties in both α and δ:

$$\min_{x\in X} \max_{\substack{\alpha\in\mathcal{B}(\hat{\alpha},\underline{\alpha},\overline{\alpha}) \\ \delta\in\mathcal{B}(\hat{\delta},\underline{\delta},\overline{\delta})}} \textstyle\sum_{(i,j)\in\mathcal{R}} \alpha_i\delta_j x_{ij}. \tag{7}$$

We say that this absolute robustness criteria uses *bilinearly weighted uncertainty* because we have a bilinear maximization problem for the inner maximization.

This is a special case of the bilinear fractional program considered by Malivert [15]. The general problem is NP-hard, but this inner bilinear program has several simplifications. The main simplification is that the polyhedron separates for the two sets of variables, and each polyhedron (the ball) is much simpler than the general case — just one equation with bounds on the variables.

In Section 5.1, we show that the inner bilinear optimization problem remains NP-hard with this special structure and even with further special structure related to sensor placement in water networks. Section 5.2 gives a straightforward algorithm that reaches a solution that need not be a global maximum. Section 5.3 gives a constant-approximation algorithm, whose error is proportional to the square of the radius of the ball (ε).

5.1 Complexity

In this section we prove that the inner bilinear optimization problem is NP-hard. We consider the restricted version of the problem:

$$\max_{\substack{\alpha \in \mathcal{P}(\hat{\alpha}, \varepsilon) \\ \delta \in \mathcal{P}(\hat{\delta}, \varepsilon)}} \sum_{(i,j)\in\mathcal{R}} \alpha_i \delta_j x_{ij}, \tag{8}$$

where both intervals have the same ε, and x satisfies the following two properties of our system:

1. The water networks we consider are directed acyclic graphs (dags) — we cannot have $(i, j) \in \mathcal{R}$ and $(j, i) \in \mathcal{R}$.
2. x satisfies transitive closure — if there is a path from i to j with no sensors ($x_{ij} = 1$) and there is a path from j to k with no sensors ($x_{jk} = 1$), then there is a path from i to k with no sensors ($x_{ik} = 1$).

We further note our special structure:

3. The domain is separable, $\mathcal{P}(\hat{\alpha}, \varepsilon) \times \mathcal{P}(\hat{\delta}, \varepsilon)$.
4. Each domain is defined by simple bounds and one constant-sum constraint.

We prove NP-hardness by reduction from the clique problem. In particular, given a graph $G = (V, E)$ with $|V| = n$ and n even, we prove that one can determine whether this graph has an $n/2$ clique by solving a bilinear program with our special structure on a transitively closed dag.

Given G, we construct a bipartite dag $G' = (A \cup P, E')$ as follows. For each vertex $v_i \in V$, create two sets of nodes and define the centers $\hat{\alpha}$ and $\hat{\delta}$:

Attack nodes. $A = \bigcup_{i=1}^{n} A_i$, where $A_i = \{a_{ij} : 1 \le j \le n\}$ for $1 \le i \le n$, with $\hat{\alpha}_{a_{ij}} = 1$ and $\hat{\delta}_{a_{ij}} = 0$ for all i, j.

Population nodes. $P = \bigcup_{i=1}^{n} P_i$, where $P_i = \{p_{ij} : 1 \le j \le n\}$ for $1 \le i \le n$, with $\hat{\alpha}_{p_{ij}} = 0$ and $\hat{\delta}_{p_{ij}} = 1$ for all i, j.

(Defining the centers in this way requires the modification of the constant-sum constraint on α to n^2, rather than 1. For the sake of keeping the notation simple, we invoke a simple scaling argument to allow this.) To build the arc set of G', for each $v_i \in V$ we add arcs forming a complete directed bipartite subgraph between the associated attack and population nodes: put $(a_{ij}, p_{ik}) \in E'$ for $j = 1, \ldots, n$ and $k = 1, \ldots, n$. These are *structural edges* that relate vertices associated with the same node in G. Further, for each edge $(v_i, v_j) \in E$, we add arcs (a_{ij}, p_{ji}) and (a_{ji}, p_{ij}) in E'. These are *graph edges* that reflect the structure of the given graph G (in which we are searching for an $n/2$ clique). There is at most one graph edge adjacent to any node in G'.

Thus, G' is a bipartite dag. Further, G' is transitively closed because all arcs go from A to P. The x of our bilinear problem is the $n^2 \times n^2$ adjacency matrix of an $n \times n$ bipartite graph. Let $M = [m_{ap}]$ be defined by $m_{ap} = 1$ if, and only if, $(a, p) \in E'$ (and $m_{ap} = 0$ otherwise) for all $a \in A$, $p \in P$. Finally, we set $\epsilon = 1 - 1/n^3$ to complete the definition of the bilinear problem.

Figure 1 shows a 6-node graph G and the constructed graph G' with 72 nodes (6 attack and 6 population per v_i for $i = 1, \ldots, 6$).

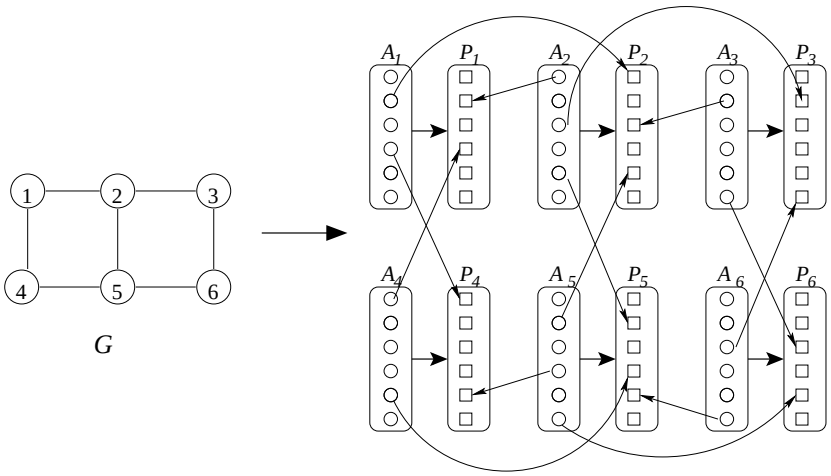

Fig. 1 Example graph with induced constructed graph.

Summarizing the construction, we have two sets of n^2 nodes each, and two sets of arcs, all directed from $a \in A$ to $p \in P$. One set of arcs (structural edges) have the form (a_{ij}, p_{ik}), so $x_{a_{ij}p_{ik}} = 1$ for all $v_i \in V$, $j, k = 1, \ldots, n$. (In figure 1 they appear as one thick arc from A_i to P_i.) The other set of arcs (graph edges) have the form (a_{ij}, p_{ji}) and (a_{ji}, p_{ij}) for $(v_i, v_j) \in E$, so $x_{a_{ij}p_{ji}} = x_{a_{ji}p_{ij}} = 1$ for all $(v_i, v_j) \in E$. For all other $f, g \in A \cup P$, $x_{fg} = 0$.

The bilinear program has the following objective value:

$$\begin{aligned}\sum_{(f,g)\in\mathcal{R}} \alpha_f \delta_g x_{fg} &= \sum_{(a,p)\in E'} \alpha_a \delta_p x_{ap} \\ &= \sum_{(a,p)\in E'} \alpha_a \delta_p \\ &= \sum_{v_i \in V} \sum_{j=1}^{n} \sum_{k=1}^{n} \alpha_{a_{ij}} \delta_{p_{ik}} \\ &\qquad + \sum_{(v_i,v_j)\in E} \left(\alpha_{a_{ij}} \delta_{p_{ji}} + \alpha_{a_{ji}} \delta_{p_{ij}}\right)\end{aligned}$$

The first equality follows from the fact that $E' = \mathcal{R} \subset A \times P$, and the second equality follows from the fact that $x_{ap} = 1$ for $(a, p) \in E'$. The α and δ values missing from the expression above (viz., α_p for $p \in P$ and δ_a for $a \in A$) are required to be zero anyway, because we defined $\hat{\alpha}_P = 0$ and $\hat{\delta}_A = 0$. We prove that an optimal solution to our bilinear program over G' with $\varepsilon = 1 - 1/n^3$ answers the question of whether G has an $n/2$ clique.

As defined above, let M be the bipartite incidence matrix for G', so its rows correspond to A and its columns to P. Because only attack nodes can have nonzero α and only population nodes can have nonzero δ, a feasible solution to the bilinear problem can redistribute α values only among the attack nodes (rows of M), and the δ values only among the population nodes (columns of M). In other words, α is constrained by $\mathcal{P}(\hat{\alpha}, \varepsilon)$ to satisfy $1-\varepsilon \leq \alpha_{a_{ij}} \leq 1+\varepsilon$ for each row a_{ij} of M. Similarly, δ is constrained by $\mathcal{P}(\hat{\delta}, \varepsilon)$ to satisfy $1 - \varepsilon \leq \delta_{p_{kl}} \leq 1 + \varepsilon$ for each column p_{kl} of M.

Given a feasible solution (α, δ) to the bilinear program, we say that a row or column of the matrix M is *selected* if respectively the α or δ value is $1+\varepsilon$. We say that an entry of the matrix is selected if both its row and column are selected. We consider only those feasible solutions that set each variable to one of its bound values, so the number of rows (columns) selected is always $\frac{n^2}{2}$ to satisfy the constant-sum constraints. This is illustrated in figure 2.

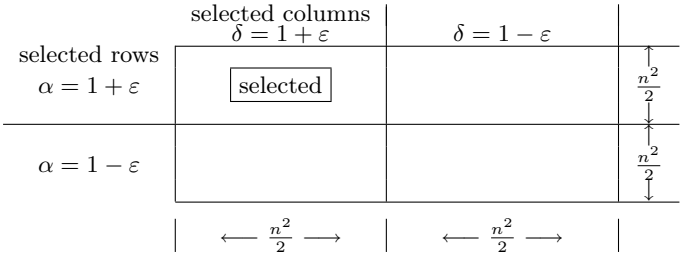

Fig. 2 Partition of M upon selecting rows, columns, and elements.

Further, we say that a vertex $v_i \in V(G)$ is selected if all rows and all columns associated with it (all $a_{ij} \in A_i$ and $p_{ik} \in P_i$) are selected. Finally, v_i is *partially-selected* if at least one row or column associated with it is selected and at least one such row or column is not selected.

In what follows, let $K = \frac{1}{2}n^3 + \frac{n}{2}(\frac{n}{2} - 1)$.

Lemma 1 *If there is an $n/2$ clique in G, the optimal value of the bilinear program is at least $K(1+\varepsilon)^2$.*

Proof Assume there is an $n/2$ clique in G. Select all the vertices in the clique, so that $\alpha_{a_{ij}} = \delta_{p_{ij}} = 1+\varepsilon$ for all v_i in the clique and all $j = 1, \ldots, n$, and $\alpha_{a_{kj}} = \delta_{p_{kj}} = 1-\varepsilon$ for all v_k not in the clique and all $j = 1, \ldots, n$. Now count the ones in M among the selected elements. Each of the $n/2$ selected vertices contributes n^2 structural edges to G', so there are $\frac{1}{2}n^3$ ones in M from the structural edges. The clique has $\frac{1}{2}\frac{n}{2}(\frac{n}{2}-1)$ edges, and each edge corresponds to two arcs (graph edges), so there are another $\frac{n}{2}(\frac{n}{2}-1)$ ones among the selected elements of M. Altogether, this gives K ones in the selected portion of M, so the value of the objective function must be at least $K(1+\varepsilon)^2$. (The other terms in the objective, corresponding to ones in the other three portions of M, have value $(1-\varepsilon)^2$ or $(1-\varepsilon)(1+\varepsilon)$, contributing a nonnegative amount.) □

Lemma 2 *There is an extreme point optimal solution to our bilinear problem in which every α and δ is $1+\varepsilon$ or $1-\varepsilon$, and both αs and δs split evenly between the two extremes.*

Proof It is already known [15] that the bilinear program has a solution at an extreme point of its domain. In our case, each extreme point is the Cartesian product of an extreme point of $\mathcal{P}(\hat{\alpha}, \varepsilon)$ and one of $\mathcal{P}(\hat{\delta}, \varepsilon)$. An extreme point of $\mathcal{P}(\hat{\alpha}, \varepsilon)$ has every variable, except at most one, at a bound value. Because n is even, the number of nodes in G' with nonzero bounds in $\mathcal{P}(\hat{\alpha}, \varepsilon)$ is even; the same applies to $\mathcal{P}(\hat{\delta}, \varepsilon)$. The constant sum constraint then requires an even split between variables at $1+\varepsilon$ and those at $1-\varepsilon$. □

Lemma 3 *If an extreme point optimal solution attains a value of at least $K(1+\varepsilon)^2$, at least K matrix elements with value 1 (edges of G') have been selected.*

Proof Assume an extreme point attains a value of at least $K(1+\varepsilon)^2$. The entire matrix has at most $n^3 + n(n-1)$ ones. Recall $K = \frac{1}{2}n^3 + \frac{n}{2}(\frac{n}{2}-1)$. So, selecting $K-1$ ones leaves at most $\frac{1}{2}n^3 + \frac{n}{2}(\frac{3n}{2}-1)+1$ unselected ones. For any $n \geq 2$, $\frac{1}{2}n^3 + \frac{n}{2}(\frac{3n}{2}-1)+1 < n^3$. Therefore, fewer than n^3 ones would be unselected, and each contributes a value of at most $(1+\varepsilon)(1-\varepsilon)$ to the objective. But, $n^3(1-\varepsilon) = 1$ by definition of ε, so the total value of the unselected ones is at most $(1+\varepsilon) < (1+\varepsilon)^2$. Finally, the bilinear value is at most

$$(K-1)(1+\varepsilon)^2 + (1+\varepsilon) < (K-1)(1+\varepsilon)^2 + (1+\varepsilon)^2 = K(1+\varepsilon)^2$$

if there are fewer than K selected ones in the extreme point. □

Theorem 2 *The bilinear problem has a maximum value of at least $K(1+\varepsilon)^2$ if, and only if, there is an $n/2$ clique in G.*

Proof Lemma 1 establishes the "if" direction. Assume that G contains an $n/2$ clique. To prove the "only if" direction, we prove that if there is an optimal

extreme point with K selected ones, there is an optimal extreme point with K selected ones and $n/2$ selected vertices. Then, the claim of the theorem follows.

Indeed, if there are K selected ones and $n/2$ selected vertices, there must be a clique: the $n/2$ selected vertices give us $n^3/2$ selected ones. That leaves no other selected rows or columns, and $\frac{n}{2}(\frac{n}{2}-1)$ selected ones from the edges, which is all the possible arcs, hence a clique.

To prove that if there is an optimal extreme point with K selected ones, there is an optimal extreme point with K selected ones and $n/2$ selected vertices, we use induction on the size of the counterexample. In other words, we consider a minimal counterexample and then show that it can be made smaller still, thus proving that a counterexample cannot exist.

We define the minimal counterexample as one with the fewest partially-selected vertices. If there are multiple such counterexamples, we take one that has a partially-selected vertex with the smallest number of selected rows plus selected columns. (Every counterexample must have at least one partially selected vertex.)

Let v_i be a partially-selected vertex with the fewest selected rows and columns.

Case 1. Suppose vertex v_i has no rows selected. Then we can find a partially-selected vertex v_j with at least one selected row and fewer than n selected columns. To see that such a v_j must exist, consider two sets: V_1, containing all v with all columns selected, and V_2, containing all v with at least one selected row. Since in an extreme point solution there are exactly $n^2/2$ selected rows and $n^2/2$ selected columns, we have $|V_2| \geq n/2$. Furthermore, v_i has at least one column selected so $|V_1| < n/2$. Therefore $V_2 \cap (V \setminus V_1) \neq \emptyset$ and we choose a v_j from this set.
We now unselect a column in vertex v_i and select a column in vertex v_j. This unselects at most one 1 (corresponding to the edge for the column originally selected), and selects at least one 1 (in the submatrix for vertex v_j), so we still have at least K selected ones. In a single row or column swap, all the newly-selected ones now contribute the greatest value of $(1+\varepsilon)^2$ to the objective (previously they contributed the middle value of $(1-\varepsilon^2)$). All unselected ones previously contributed the greatest value and now contribute the middle value. Therefore, as long as the number of selected ones after the swap does not decrease, the objective function also does not decrease. Thus, we obtain a smaller counterexample.

Case 2. Symmetrically, if the minimal vertex v_i has no selected columns, swap a row associated with a partially-selected vertex with at least one column selected and room to select another row. Therefore, the minimal vertex has at least one row and one column selected.

Case 3. Suppose the minimal vertex v_i has $r_i \geq 1$ rows selected and $c_i \geq 1$ columns selected. Suppose there is a partially-selected vertex v_j with $r_j > r_i$ selected rows. If v_j has fewer than n selected columns, we can swap a column from v_i to v_j and get a smaller counterexample. If v_j has n selected columns, swapping a row from v_i to v_j again gives a smaller counterexample provided $c_i < n$. If $c_i = n$, then r_i is the smallest among all partially-selected vertices (that is, all other partially-selected vertices

v_j have $r_j \geq r_i$). There must be at least $n/2 + 1$ partially-selected or selected vertices. These cannot all have n selected columns. Therefore, there exists a vertex v_j with $c_j < n$. We must also have $r_j > r_i$ because if $r_j = r_i$ and $c_j < n$, then v_j would be the minimal vertex (recall $c_i = n$). A similar argument holds if there is a v_j with more than c_j columns selected. Therefore, in any counterexample all partially-selected vertices have the same number of rows and columns selected as our minimal vertex v_i, say $n > k \geq 1$ rows and $n > \ell \geq 1$ columns. We know $k < n$ because otherwise (if $k = n$), there are exactly $n/2$ vertices with all rows selected and exactly $n/2$ vertices with no row selected. Since no partially-selected vertex can have zero rows selected (from Case 1), we can select only those columns corresponding to the $n/2$ vertices that have a row selected. Because we must select $n^2/2$ columns, we must select all columns for all these vertices and in fact no vertices are partially selected. A similar argument shows that $\ell < n$.

Pick one partially-selected vertex v_j, unselect a row and a column of v_i, and select at the same time a row and a column of v_j. Consider the submatrices M_{v_i} and M_{v_j} of M, defined by the rows and columns of v_i and, respectively, v_j. By unselecting a row of v_i, we unselect ℓ ones in M_{v_i}, and unselect at most one edge. Then, unselecting the column unselects at most $k - 1$ ones in M_{v_i}, and unselects at most one edge. When we select a row of v_j, we select ℓ ones in M_{v_j}. When we select a column of v_j, we select $k+1$ ones (including the 1 for the new row and column) in M_{v_j}. So, we have unselected at most $k + \ell + 1$ ones, and selected at least $k + \ell + 1$ ones, yielding a smaller counterexample.

We have shown that in all cases, it is possible to reduce our counterexample to a smaller one with at least as many ones. This implies that there is no counter-example, and thus proves the theorem. □

We have thus established that our inner bilinear optimization problem is NP-hard despite the simple domain. Our proof used ε arbitrarily close to 1. In practice, we generally have $\varepsilon \leq \kappa$ for some $\kappa < 1$, such as $\kappa = \frac{1}{4}$. In such a case our proof does not apply, and the NP-hardness remains an open question. Further, this is a theoretical result, and it remains to examine some real applications to determine how difficult it is to solve this problem in practice (see §6).

5.2 Alternating Ascent

Although we have shown that the inner bilinear program (7) is NP-hard, we may still need to solve this in a practical manner. We consider a heuristic search strategy for solving the inner bilinear program for a given set of x values, which could be used as an inner loop for a general-purpose search strategy (e.g., meta-heuristic methods).

Figure 3 describes a simple local search strategy for solving the bilinear program for given values of x. Given x, one way to seek a solution is to alternate between α and δ, solving an LP in each iteration. Each maximization is a linear program, the same as the inner maximization of the linearly

weighted case. Since each instance has different weights, we may need to sort each time. After a finite number of iterations, we terminate with extreme point optimal solutions to each polytope.

Alternating Ascent Algorithm

0. Initialize. Choose $\alpha^0 \in \mathcal{B}(\hat{\alpha}, \underline{\alpha}, \overline{\alpha})$, and set $k = 0$.
1. Solve for $\boldsymbol{\delta}$. Given α^k, compute

$$\delta^k \in \operatorname{argmax} \left\{ \textstyle\sum_{(i,j)\in\mathcal{R}} \alpha_i^k \delta_j x_{ij} \ : \delta \in \mathcal{B}(\hat{\delta}, \underline{\delta}, \overline{\delta}) \right\}$$

2. Solve for $\boldsymbol{\alpha}$. Given δ^k, compute

$$\alpha^{k+1} \in \operatorname{argmax} \left\{ \textstyle\sum_{(i,j)\in\mathcal{R}} \alpha_i \delta_j^k x_{ij} \ : \alpha \in \mathcal{B}(\hat{\alpha}, \underline{\alpha}, \overline{\alpha})) \right\}$$

3. Increment $k \leftarrow k + 1$ and repeat steps 1–3 until $\delta^k = \delta^{k-1}$ ($k > 0$, step 1) or $\alpha^{k+1} = \alpha^k$ (step 2).

Fig. 3 An alternating ascent algorithm for the bilinear program (7), for a given value of x.

The bilinear program has an optimal solution among the extreme points of $\mathcal{B}(\hat{\alpha}, \underline{\alpha}, \overline{\alpha})$ and $\mathcal{B}(\hat{\delta}, \underline{\delta}, \overline{\delta})$. This follows from elementary theory of LP and noting a necessary condition for (α^*, δ^*) to be optimal [15]:

$$\alpha^* \in \operatorname{argmax} \left\{ \textstyle\sum_{(i,j)} \alpha_i \delta_j^* x_{ij} \ : \alpha \in \mathcal{B}(\hat{\alpha}, \underline{\alpha}, \overline{\alpha}) \right\}$$

$$\delta^* \in \operatorname{argmax} \left\{ \textstyle\sum_{(i,j)} \alpha_i^* \delta_j x_{ij} \ : \delta \in \mathcal{B}(\hat{\delta}, \underline{\delta}, \overline{\delta}) \right\}.$$

Unfortunately, this is not sufficient. If we terminate the algorithm with (α^*, δ^*), all we can say is that this satisfies the above necessary condition. We cannot rule out the possibility that a simultaneous change in α and δ would increase the objective value. We call a point that satisfies these necessary conditions an *alternating ascent solution*.

An alternating ascent solution is a KKT point (i.e., satisfies the first-order, Karush-Kuhn-Tucker conditions for optimality). Further, if the algorithm goes one iteration, the alternating ascent solution cannot be a minimum. It can, however, be a saddle point.

We have found examples where the alternating ascent solution is not a global maximum. Although the objective can not be improved by changing α or δ, keeping the other fixed, we were able to increase the objective value with a simultaneous change in α and δ (just to neighboring extreme points of their respective polyhedra). We ran some experiments to see how often this occurs, and the particular runs indicate that alternating ascent gets trapped at a non-global-maximum point from relatively few starting points. In all cases, an we found an improvement by combining neighbors of α in $\mathcal{B}(\hat{\alpha}, \underline{\alpha}, \overline{\alpha})$ with neighbors of δ in $\mathcal{B}(\hat{\delta}, \underline{\delta}, \overline{\delta})$. While inconclusive, this indicates that it may be possible to identify conditions under which such a search procedure would

reach the global maximum. The extreme points of the bilinear program are easy to characterize, so it may be possible to exploit this information in some cases to guide the local search. For example, it might be possible to jump to a non-adjacent extreme point to avoid terminating at a solution that is not a global maximum.

5.3 Linear Programming Relaxation

In this section we describe a linear relaxation of the bilinear case, using McCormick's bounds, commonly used by the αBB method [2]. McCormick's bounds have been used to approximate a bilinear form, $u^{\mathsf{T}}v$, on a rectangle $[\underline{u}, \overline{u}] \times [\underline{v}, \overline{v}]$ with linear functions using the simple inequalities [2,18]:

$$(u - \underline{u})^{\mathsf{T}}(v - \underline{v}) \geq 0, \quad (u - \overline{u})^{\mathsf{T}}(v - \overline{v}) \geq 0$$

$$(u - \underline{u})^{\mathsf{T}}(v - \overline{v}) \leq 0, \quad (u - \overline{u})^{\mathsf{T}}(v - \underline{v}) \leq 0.$$

These yield linear (affine) functions that bound $u^{\mathsf{T}}v$ from below and above. In our case, since we are maximizing, we use only the upper bounds in the following LP relaxation which we call the *McCormick Approximation*:

$$\begin{aligned} &\max z \,:\, \alpha \in \mathcal{B}(\hat{\alpha}, \underline{\alpha}, \overline{\alpha}),\ \delta \in \mathcal{B}(\hat{\delta}, \underline{\delta}, \overline{\delta}), \\ &z - \textstyle\sum_{(i,j)\in\mathcal{R}} \left(\alpha_i\, \underline{\delta}_j + \overline{\alpha}_i\, \delta_j\right) x_{ij} \leq -\sum_{(i,j)\in\mathcal{R}} \overline{\alpha}_i\, \underline{\delta}_j\, x_{ij} \\ &z - \textstyle\sum_{(i,j)\in\mathcal{R}} \left(\alpha_i\, \overline{\delta}_j + \underline{\alpha}_i\, \delta_j\right) x_{ij} \leq -\sum_{(i,j)\in\mathcal{R}} \underline{\alpha}_i\, \overline{\delta}_j\, x_{ij} \end{aligned}$$

More generally, the McCormick Approximation has a guaranteed error of $\frac{1}{4}(\overline{\alpha} - \underline{\alpha})^{\mathsf{T}}(\overline{\delta} - \underline{\delta})$ times the optimal, established by Androulakis, et al. [2] for just the rectangle, $[\underline{\alpha}, \overline{\alpha}] \times [\underline{\delta}, \overline{\delta}]$. It is easy to extend this to account for only the contaminated nodes, giving an approximation guarantee of $\frac{1}{4}(\overline{\alpha} - \underline{\alpha})^{\mathsf{T}} x\, (\overline{\delta} - \underline{\delta})$. For example, if the uncertainties are proportional to epsilon ($\overline{\alpha} = (1+\varepsilon)\hat{\alpha}$, $\underline{\alpha} = (1-\varepsilon)\hat{\alpha}$, $\overline{\delta} = (1+\varepsilon)\hat{\delta}$ and $\underline{\delta} = (1-\varepsilon)\hat{\delta}$) then it follows that we have a $1 + \varepsilon^2$ approximation.

6 Preliminary Computational Study

In this section we report preliminary computational results for sensor placement problems on two water networks. We explore the computational difficulty of solving the full sensor placement problem, which we can solve for the versions that involve MILPs. We also explore the difficulty of computing optimal and approximate solutions to the bilinear inner optimization problem for the case with uncertainty in both attack weights and population distribution.

We performed computations on two real water networks with 97 and 470 nodes respectively. Water networks for large cities have tens of thousands of nodes, so these test cases are still about 2 orders of magnitude smaller

than many important sensor placement problems. However, these experiments serve to illustrate performance difference even for problems of this modest scale.

When we described the sensor placement problems earlier, we assumed only one flow pattern. Most cities, however, have multiple normal demand patterns cycling throughout each day. For example, desert cities may recharge tanks at night and drain them during the day. Our water networks have four daily demand patterns, which are used in our experiments. This makes the computations more realistic and more difficult. Sensor placements must protect against attacks that can have fundamentally different effects based upon when they occur. The only changes required to the models is to add an extra dimension to the input data. Attack weights and population are now associated with each of the four patterns, and the expectation in the objective is now taken over these patterns as well.

6.1 Linearly Weighted Uncertainties

Our first set of computational results considers the linearly weighted robust formulation, which can be modeled as a MILP. Our "base case" is the *central value model*, where we assume attack weights and population values are exactly equal to their central values (no robustness modeling). Our experiments illustrate how much additional computational effort is required to obtain a robust solution. We proved that the incremental effort is zero for the unweighted robust model, and we include computational results for two linearly weighted cases:

1. attack weights are uncertain and populations are fixed at their central values.
2. attack weights are fixed at their (non-uniform) central values and populations are uncertain.

In all experiments, we consider uncertainties that are proportional to the central value.

Table 1 shows the total computation time for the central value model (base case) and for robustly solving the sensor placement problem for our two example networks. The time reported in the table is the CPU seconds required to solve each problem running on a dual 3.06 GHz Intel® Xeon™ processor, Linux 2.6.10 system with 2 GB RAM. The number of nodes, reported just after the time, is the size of the search tree computed by CPLEX®.

Table 1 Computational Results for Complete Sensor Placement Solutions[†]

Network						Linearly Weighted			
				Central Value		Population		Attack weights	
nodes	arcs	N_s	ϵ	time	nodes	time	nodes	time	nodes
97	234	5	0.1	21.9	16	188.8	24	135.9	20
			0.2			50.0	93	25.2	54
			0.3			57.1	218	30.7	52
		10	0.1	2.9	0	15.0	0	17.7	11
			0.2			27.7	14	20.1	3
			0.3			32.7	10	20.2	3
		20	0.1	11.9	4	28.5	0	27.1	0
			0.2			197.7	9	38.7	0
			0.3			248.1	29	57.9	40
470	1198	5	0.1	165.5	0	145.1	0	504.2	0
			0.2			463.5	0	546.7	0
			0.3			450.8	0	479.6	0
		10	0.1	69.9	2	678.7	2	549.9	2
			0.2			790.3	9	617.8	2
			0.3			892.5	13	764.2	4
		20	0.1	69.8	21	75.4	13	1472.9	63
			0.2			1335.6	27	1441.1	44
			0.3			2482.0	165	1722.8	101

[†]These were solved with AMPL® and CPLEX.

These results suggest that computational difficulty will generally increase with the number of sensors, N_s, and the robustness tolerance, ε. The robust formulation is almost always more difficult to solve than the central value model. Furthermore, the results with the larger network show that this robust formulation requires over an order of magnitude more time to solve in many cases.

6.2 Bilinearly Weighted Uncertainty

Although we have shown that just the bilinear subproblem, itself, is NP-hard, this result provides limited information about the practical comptuational difficulty of this formulation. Unfortunately, no currently available system can solve the bilinearly weighted formulation with global optimality confirmed. Recall that this model has the form:

$$\min_{x \in X} \max_{\alpha,\delta} \alpha^{\mathsf{T}} x \, \delta$$

The minimand is a maximum, and no system can solve this at present. Although methods like BARON [19] can solve the inner bilinear program with confirmed optimality, we must combine that with an outer search strategy on x. Designing and implementing an outer search coupled with BARON is beyond the scope of this paper. The full sensor placement model could be written as a semi-infinite program by replacing the maximand with z and adding the infinite number of constraints:

$$z \geq \alpha^{\mathsf{T}} x \, \delta \text{ for all } \alpha, \delta.$$

Table 2 Computational Results for Bilinear Model for a Fixed Sensor Placement[†]

Network	N_s	Central value	ϵ	Alternating Ascent	McCormick Approx	Global Value	Time
1	5	4100	0.1	4949	4949	4949	49.91
			0.2	5878	5878	5878	65.43
			0.3	6887	6887	6887	68.51
	10	3955	0.1	4777	4777	4777	37.87
			0.2	5677	5677	5677	37.92
			0.3	6654	6654	6654	38.35
	20	3907	0.1	4721	4721	4721	27.94
			0.2	5613	5613	5613	24.76
			0.3	6582	6581	6582	22.49
2	5	134.54	0.1	158.62	158.61	158.62	2848.73
			0.2	184.70	184.67	184.70	2953.75
			0.3	212.81	212.70	212.81	2765.20
	10	98.12	0.1	115.70	115.70	115.70	2289.35
			0.2	134.75	134.72	134.75	2285.12
			0.3	155.30	155.19	155.30	2275.44
	20	70.13	0.1	82.55	82.54	82.55	1908.85
			0.2	95.98	95.97	95.98	1901.59
			0.3	110.46	110.40	110.46	1910.16

[†]Alternating Ascent and McCormick Approximation were computed within MATLAB® (m files available upon request). McCormick Approximation used Mosek [1] to solve the LP. Global maxima were computed by BARON [19,20]. The time reported is the CPU seconds for BARON to compute the global maximum; the approximations took no more than a few seconds each.

However, no currently available system can solve this model either.

Consequently, our computational experiments have focused on the time required to solve the bilinear subproblem for a fixed sensor placement. Table 2 compares the value of the central value model with the values of solutions for the bilinear subproblem. This subproblem is solved with the alternating ascent heuristic, with the exact BARON solver, and it is approximated with the McCormick approximation method. In these experiments, we set the contamination array (x) for the bilinear experiments using the values from the central value MILP model.

In all of these experiments, the BARON solver is one or more orders of magnitude slower than the other methods, which should be expected because it is confirming global optimality. Remarkably, the alternating ascent heuristic generates solutions with the same value in every case. This contrast provides strong evidence that it is worth exploring the application of these heuristics for assessing the robustness of solutions with bilinearly weighted uncertainties. The approximation method was less effective than the alternating ascent heuristic overall, though the value of the solution that it generates was often quite similar. Finally, it is noteworthy that the runtime for BARON to solve the bilinear subproblem was often at least as long as the cost of solving the linearly weighted models. This suggests that a solver that can exactly

solve the entire bilinearly weighted model will be significantly more expensive to solve than the MILP models for the linearly weighted uncertainties.

7 Discussion

There are many possible objectives for sensor placement that reflect various costs and risks of an attack on a network [21]. Previous work has considered problem formulations that minimize the volume of water consumed before detection [11], minimize the time to detection [13], and minimize the population exposed to contaminants before detection [4, 14]. This paper presents a foundation upon which these objectives, taken separately or multiply, can be considered in a manner that addresses data uncertainties.

Robust optimization addresses a need to hedge against uncertainty, and these uncertainties are a fundamental property of sensor placement problems. Data like attack weights and population distribution are based on expert judgement and incomplete source data. Furthermore, these data are expected to vary during the years after sensors are deployed in an early warning system. Although a number of criteria for robust optimization have been studied, the interval data model, with a constant-sum constraint, fits these sensor placement problems well.

The simplest case that we have considered is the unweighted uncertainty model, which represents two cases: (1) only the attack weights are uncertain, and the objective is the expected number of nodes that are contaminated without detection; and, (2) the attack weights are uniform, and the objective is the expected population that become contaminated without detection. In this case, if the interval is restricted to a fixed proportion of the central vector, we have shown this problem has a permanent solution. This is counter-intuitive, as data uncertainties ***should*** affect our decision. However, we have proven that robust solutions can be obtained from just the central value (which may be the most likely realization). Thus, obtaining a robust solution does not make the computation more difficult.

The next level of difficulty is the linearly weighted case: one set of parameters is fixed at their central values, while the other is uncertain. If we let the population be uncertain, the attack weights are presumed non-uniform; otherwise, the model reduces to the unweighted case. In our preliminary experiments, the linearly weighted case added a modest amount of computational effort to solve the overall problem.

Although the linearly weighted case can address robustness issues, the bilinear model is the most general by allowing uncertainty around non-uniform central values of both attack weights and population. We proved that the maximization subproblem is NP-hard, even with the simplifications of being in a ball with fixed-proportionate bounds and one constant-sum constraint. However, it is unclear whether this maximization subproblem remains NP-hard if we require that $\varepsilon \leq \kappa$ for some $\kappa < 1$. Our NP-hardness proof requires ε to be arbitrarily close to 1. Further, our computational experiments suggest that we can obtain solutions for the bilinear subproblem, or at least provide bound information in solving the master problem. However, we expect that

it will be very difficult to compute exact solutions for bilinearly weighted formulations.

This paper has focused on modeling robustness and analyzing robust formulations. Our computational experiments are preliminary, and thus we cannot make strong predictions with them. A more comprehensive computational study of these techniques is clearly an important avenue of future research, including the development of computational techniques for exactly solving the bilinearly weighted formulation.

Acknowledgements Sandia National Laboratories is a multipurpose laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000. We thank Nick Sahinidis and GAMS Development Corp. for letting us use GAMS/BARON for the computational study. Nick Sahinidis also provided very helpful guidance on the effective use of BARON for our nonlinear problems. Finally, we thank the two referees for their extensive comments that led to a better presentation and their bringing the relevance of the McCormick bounds to our LP relaxation (§5.3, p. 16).

References

1. Anderson, E.: The MOSEK Optimization Toolbox for MATLAB Version 2.5, User's Guide and Reference Manual. World Wide Web, http://www.mosek.com (1999–2002)
2. Androulakis, I., Maranas, C., Floudas, C.: αBB: A global optimization method for general constrained nonconvex problems. Journal of Global Optimization **7**, 337–363 (1995)
3. Berry, J., Hart, W.E., Phillips, C.A., Uber, J.: A general integer-programming-based framework for sensor placement in municipal water networks. In: Proceedings of the World Water and Environment Resources Conference (2004)
4. Berry, J.W., Fleischer, L., Hart, W.E., Phillips, C.A., Watson, J.P.: Sensor placement in municipal water networks. Journal of Water Resources Planning and Management **131**(3), 237–243 (2005)
5. Bertsimas, D., Sim, M.: Robust discrete optimization and network flows. Mathematical Programming Series B **98**, 49–71 (2003)
6. Bertsimas, D., Sim, M.: The price of robustness. Operations Research **52**, 35–53 (2004)
7. Dantzig, G.B.: Linear programming under uncertainty. Management Science **1**(3, 4), 197–206 (1955)
8. Ferson, S., Joslyn, C.A., Helton, J.C., Oberkampf, W.L., Sentz, K.: Challenge problems: Uncertainty in system response given uncertain parameters. Reliability Engineering and System Safety **85**, 11–19 (2004)
9. Greenberg, H.: Mathematical Programming Glossary. World Wide Web, http://www.cudenver.edu/~hgreenbe/glossary/ (1996–2005)
10. Hoffman, F.O., Hammonds, J.S.: Propagation of uncertainty in risk assessments: The need to distinguish between uncertainty due to lack of knowledge and uncertainty due to variability. Risk Analysis **14**(5), 707–712 (1994)
11. Kessler, A., Ostfeld, A., Sinai, G.: Detecting accidental contaminations in municipal water networks. Journal of Water Resources Planning and Management **124**(4), 192–198 (1998)
12. Kouvelis, P., Yu, G.: Robust Discrete Optimization and Its Applications. Kluwer Academic Publishers, Norwell, MA (1996)
13. Kumar, A., Kansal, M.L., Arora, G.: Discussion of 'detecting accidental contaminations in municipal water networks'. Journal of Water Resources Planning and Management **125**(4), 308–310 (1999)
14. Lee, B., Deininger, R.: Optimal locations of monitoring stations in water distribution system. Journal of Environmental Engineering **118**(1), 4–16 (1992)

15. Malivert, C.: An algorithm for bilinear fractional problems. Tech. Rep. 1998-03, Université de Limoges, Cedex, France (1998)
16. Oberkampf, W.L., DeLand, S.M., Rutherford, B.M., Diegert, K.V., Alvin, K.F.: Error and uncertainty in modeling and simulation. Reliability Engineering and System Safety **75**, 333–357 (2002)
17. Rowe, W.D.: Understanding uncertainty. Risk Analysis **14**(5), 743–750 (1994)
18. Ryoo, H.S., Sahinidis, N.V.: Global optimization of nonconvex NLPs and MINLPs with applications in process design. Computers & Chemical Engineering **19**(5), 551–566 (1995)
19. Sahinidis, N., Tawarmalani, M.: BARON 7.2.5: Global Optimization of Mixed-Integer Nonlinear Programs, User's Manual (2005)
20. Tawarmalani, M., Sahinidis, N.: Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. Mathematical Programming **99**, 563–591 (2004)
21. Watson, J.P., Greenberg, H.J., Hart, W.E.: A multiple-objective analysis of sensor placement optimization in water networks. In: Proceedings of the World Water and Environment Resources Congress. American Society of Civil Engineers (2004)
22. Yaman, H., Karasan, O.E., Pinar, M.C.: The robust minimum spanning tree problem with interval data. Operations Research Letters **29**, 31–40 (2001)
23. Yaman, H., Karasan, O.E., Pinar, M.C.: Restricted robust optimization for maximization over uniform matroid with interval data uncertainty. Tech. rep., Bilkent University Technical Report (2004)