A STOCHASTIC PROGRAMMING APPROACH FOR GAS DETECTOR PLACEMENT IN PROCESS FACILITIES

S.W. Legg[1], J.D. Siirola[2], J.P. Watson[2], S.G. Davis[3], Are Bratteteig [3], C.D. Laird[1,⋆]
1) Department of Chemical Engineering, Texas A&M University
3122 TAMU, College Station, TX 77843
2) Discrete Algorithms and Math Department, Sandia National Laboratories
Mail Stop 1318, P.O. Box 5800, Albuquerque, NM 87185
3) GEXCON, Bethesda, MD

Abstract

Given hundreds of potential gas detector locations, a stochastic programming formulation is developed for determining the optimal placement of these sensors for detecting gas release events in petrochemical facilities. Using a rigorous dispersion model with actual geometry from the process facility, hundreds of different scenarios are simulated using FLACS with different leak locations, process conditions, and weather properties. Pyomo, a python-based optimization package, is used to formulate the multi-scenario, mixed-integer programming problem. Using CPLEX to solve the formulations, different objective functions are explored. Optimal results are presented for the minimum number of sensors required to detect all scenarios, the minimum expected time to detect events using a fixed number of sensors, and a robust formulation that minimizes the maximum time to detection across all scenarios. In all these examples, the formulation can be solved efficiently for real, large-scale problems.

Keywords

Gas Leak Detection, Process Safety, Sensor Placement, Stochastic Programming, Mixed-Integer Linear Programming

Introduction

Gas detection, specifically the detection of combustible and toxic gas release events, is a key component of modern process safety. Combustible gas detection relies upon the detection of a gas before it reaches either its lower explosive or lower flammable limit, LEL and LFL, respectively. These limits refer to the gas concentrations at which a dispersed gas cloud in air will allow a flame front to spread when exposed to an ignition source. Toxic gas detectors are designed to detect and alarm at a concentration related to the impact on human health, typically expressed in parts per million (ppm). Usually, combustible gas detectors and toxic gas detectors are calibrated to alarm when they detect some fixed fraction of the LEL, LFL, or toxic gas limits.

In addition to proper selection of the sensor specifications needed for a particular plant, appropriate placement of these sensors is required to ensure effective detection of likely gas release events. Improper placement of gas detectors may reduce the probability of detecting a particular release, or even yield a sensor completely useless. Given complex process geometries and varying weather conditions, the best placement of gas sensors is not obvious. In this paper, we present a stochastic programming formulation for determining the optimal plant-specific placement of gas detectors. FLACS, a modern package for explosion and dispersion modeling, is used to simulate hundreds of process specific leak scenarios. Armed with scenarios from the rigorous dispersion model, a multi-scenario, mixed-integer linear programming (MILP) formulation is developed to perform optimal sensor placement over several different objectives. This problem formulation is used to determine the minimum number of sensors required to detect all release scenarios. With small adjustments to the problem formulation, we can solve for the sensor placement that minimizes the expected time to detection across all scenarios, as well as a robust optimiza-

⋆Corresponding author: carl.laird@tamu.edu

tion formulation that minimizes the maximum time to detection. Furthermore, since the approach is very efficient, we can solve the formulation repeatedly, varying the number of available sensors, and show the overall effectiveness of increasing the number of sensors in a plant.

Background material is discussed first. Next, the multi-scenario MILP formulation for the optimal placement of gas sensors is given. Using this formulation, along with slight modifications, numerical results on a real process geometry are presented. In closing, we review the presented material and offer conclusions and ideas for continuing research.

Background

Two types of detectors are commonly used for the detection of combustible gas clouds: catalytic and infrared sensors. Catalytic gas sensors detect the presence of a chemical contaminant by an oxidation reduction reaction with the catalyst. These sensors provide accurate contaminant concentrations in air and have a low unit cost, however, they are susceptible to catalyst poisoning and will not reveal a sensor failure without regular maintenance. Infrared sensors work by detecting the amount of infrared energy absorbed by a contaminant cloud at specific wavelengths. These sensors possess a higher unit cost and lower overall accuracy in terms of concentration quantification, but can often detect contaminant over a large area and require less maintenance.

For toxic release events, electrochemical or semiconductor gas detectors are typically used. Electrochemical detectors allow a gas to diffuse through a porous membrane to an electrode where it can either be oxidized or reduced to determine concentration. A semiconductor detector detects a gas by changes in resistance on a conductive surface in the presence of a contaminant. Both detector types are calibrated to alarm at a chemical-specific threshhold concentration. All of the sensors mentioned can be used in either a fixed or portable form; however for the purposes of this paper, all sensors are considered fixed.

The placement of gas detection sensors in the petrochemical industry, whether for toxic or flammable releases, is typically based upon heuristics or "rules of thumb" which rely on identifying equipment of interest and the properties of the gas in question. Sensors are then placed near areas where leaks are expected to originate (UK Health & Safety Executive, 2011). However, given several uncertainties (e.g., leak location, process conditions, and weather conditions), we seek to develop a stochastic programming framework that rigorously considers these uncertainties when determining the sensor placement.

The work presented in this paper is an extension of a mixed-integer programming based optimization approach developed by Berry et al. (2005) for the optimal placement of contamination sensors in large-scale water distribution networks. By precalculating a large set of possible contamination scenarios, point sensors can be placed throughout the network in order to minimize the effects of a contamination event subject to a set of cost constraints. Extensions to this work were made to relax constraints that required each event to be detected by at least one sensor with the addition of an unphysical "dummy" location (Berry et al., 2005). Additionally, further extensions considered effects of time on the sensor placement (including time to detection) (Berry et al., 2006). In our work, we will extend this stochastic programming formulation to determine optimal placement of gas detectors in process facilities.

Berry et al. (2005) assumed that point sensors detected contamination if and only if the contaminant physically passed through the sensor. In this sense, the sensors used in their formulation exhibit the same properties possessed by infrared and catalytic point gas sensors used in the petrochemical industry (Fire & Safety World Online, 2011). Additionally, the industry utilizes line-of-sight infrared sensors which can detect a contaminant crossing the infrared beam over long distances. These sensors are not as effective at determining the size or concentration of a contaminant cloud, but are extremely adept at detecting the existence of a release and also fail in a manner which triggers a false positive. This attribute lowers the probability of a missed detection. The water sensor network problem and the open-air gas dispersion problem differ mainly in the simulation frameworks used to create the possible contamination (or gas leak) scenarios. An adaptation of the sensor placement formulation to the open air problem has been proposed for homeland security purposes by Hamel et al. (2006). In their work, computational fluid dynamics (CFD) is incorporated to generate a set of potential urban attack events where a nuclear, biological, or chemical agent is dispersed into air. This CFD modeling is a corollary to the EPANET water network model used in the work by Berry et al. (2006). In our work, CFD software is used to generate rigorous dispersion simulations for various leak locations, conditions, and weather variables. The event scenarios for this paper were generated by GexCon using their CFD software, FLACS (GexCon, 2011). These simulations were based on flammable gas dispersion using a real process geometry.

Sensor Placement Formulation

The sensor placement formulation for this paper is built upon the work previously developed by Berry et al. (2006). This formulation considers a generic, represen-

Table 1. Tables of Symbols

Symbol	Meaning
$L = \{l_1, l_2, l_3, ..., l_N\}$	Set of candidate sensor locations
N	Number of candidate sensor locations
l	Sensor location index
$A = \{a_1, a_2, a_3, ..., a_M\}$	Set of release events
M	Number of release events
a	Release event index
$\mathcal{L}_a$	Candidate sensor locations affected by event a
α_a	Probability that event a occurs
$D_{a,i}$	Damage coefficient for event a at location i
p	Maximum number of sensors allowed
s_l	Binary variable for sensor placement
$x_{a,i}$	Variable for the sensor at location i that first detects event a
q	Dummy sensor location for undetected events
t_{max}	Maximum time to detect

tative chemical or refining facilty. In this facility, there exists a set of potential gas sensor locations, defined as

$$L = \{l_1, l_2, l_3, ..., l_N\} \tag{1}$$

where L is the set of all N potential locations, indexed by l. Also, the set of potential release events is defined as

$$A = \{a_1, a_2, a_3, ..., a_M\} \tag{2}$$

where A is the set of all M release scenarios, indexed by a. It is important to note that not all sensor locations may be affected by a given release scenario a. Therefore, a subset of L can be defined for each release event $a \in A$, defined here as $\mathcal{L}_a$, that represents the sensor locations that are affected by the particular release event a. The parameter α_a is the probability associated with release event a out of all the events in A. That is, the α_a parameters are set such that,

$$\sum_{a \in A} \alpha_a = 1. \tag{3}$$

A damage coefficient $D_{a,i}$ is identified for each release event $a \in A$ and each sensor location $i \in \mathcal{L}_a$. This coefficient represents the amount of damage that will be done if event a is first detected by a sensor at a location i. The physical meaning of the damage coefficient is determined by the particular problem at hand. If the released material is a flammable or explosive threat for example, $D_{a,i}$ can be represented as the total volume of gas that lies within the flammable region of concentrations for that material. Similarly, in the event of a toxic gas release, it may be more applicable to define $D_{a,i}$ as the volume of gas above some toxic threshhold, such as the LC_{50}. It should be noted that this damage coefficient can be tailored to represent a variety of different release types.

The binary decision variable s_l is used in the formulation to represent the existence of a sensor, $s_l = 1$, or the lack of a sensor, $s_l = 0$, at location l. The continuous decision variable $x_{a,i}$ is used to indicate the first sensor $i \in \mathcal{L}_a$ to detect a particular event a. While $x_{a,i}$ is a continuous variable, because of the problem formulation, each $x_{a,i}$ will be forced to be either 0 or 1 at the solution (Berry et al., 2006). The variable $x_{a,i}$ will be 1 if sensor location i is the first to detect scenario a, and 0 otherwise. Finally, because gas sensors can be a cost prohibitive item, a parameter p can be assigned limiting the total number of sensors to be placed.

The overall mixed-integer formulation is written as

$$\min \quad \sum_{a \in A} \alpha_a \sum_{i \in \mathcal{L}_a} D_{a,i} x_{a,i} \tag{4a}$$

subject to:

$$\sum_{i \in \mathcal{L}_a} x_{a,i} = 1 \qquad \forall a \in A \tag{4b}$$

$$x_{a,i} \leq s_i \qquad \forall a \in A, i \in \mathcal{L}_a \tag{4c}$$

$$\sum_{l \in L} s_l \leq p \tag{4d}$$

$$s_l \in \{0, 1\} \qquad \forall l \in L \tag{4e}$$

$$0 \leq x_{a,i} \leq 1 \qquad \forall a \in A, i \in \mathcal{L}_a. \tag{4f}$$

The first constraint specifies that each event must have a sensor that first detects that event. However, this constraint can be easily relaxed, and is in our problem formulation, by including a dummy location q in $\mathcal{L}_a$ that represents a situation where the event a is not detected (Berry et al., 2006). A corresponding damage value $D_{a,q}$ is assigned for the situation where event a is not detected. Typically, this value is selected as the maximum resulting release size without mitigation or detection. The second constraint forces a sensor, $s_i = 1$, to be located at any location $i \in \mathcal{L}_a$ that is the first to detect, described by $x_{a,i} = 1$. Therefore, if $s_i = 0$, that location cannot claim to be the first to detect an event. The third constraint places a limit p on the total number of sensors placed in the facility. Finally, the decision variables are defined as binary and continuous on [0,1] for s_l and $x_{a,i}$, respectively. The objective function of this

formulation seeks to minimize the expected value of the damage over the set of event scenarios.

This formulation can be modified to find the minimum number of sensors required to detect all events. The objective function can be rewritten as,

$$\min \quad p \tag{5}$$

and the dummy location q is removed, in order to force detection of every event.

Another formulation provides the so-called robust optimization formulation. That is, considering the damage resulting from each potential event, this formulation seeks to find a sensor placement that minimizes the maximum damage over all scenarios. The objective function for this formulation can be written as,

$$\min \quad \max_{a \in A} \sum_{i \in \mathcal{L}_a} D_{a,i} x_{a,i}. \tag{6}$$

This min-max objective function can be easily reformulated into a standard mixed-integer linear problem through the addition of one new variable t_{max} and a set of constraints, giving the following formulation,

$$\min \quad t_{max} \tag{7a}$$

subject to:

$$\sum_{i \in \mathcal{L}_a} D_{a,i} x_{a,i} \leq t_{max} \qquad \forall a \in A \tag{7b}$$

$$\sum_{i \in \mathcal{L}_a} x_{a,i} = 1 \qquad \forall a \in A \tag{7c}$$

$$x_{a,i} \leq s_i \qquad \forall a \in A, i \in \mathcal{L}_a \tag{7d}$$

$$\sum_{l \in L} s_l \leq p \tag{7e}$$

$$s_l \in \{0,1\} \qquad \forall l \in L \tag{7f}$$

$$0 \leq x_{a,i} \leq 1 \qquad \forall a \in A, i \in \mathcal{L}_a. \tag{7g}$$

The three formulations presented above will be used in the next section to provide optimal detector placement for a real process facility.

Placement Results

The mixed-integer problems presented in this paper were formulated and solved using Pyomo, a package from the Coopr open-source software library (Hart et al., 2011), which is part of the COIN-OR project. The Python Optimization Modeling Objects (Pyomo) software package utilizes the Python language to develop and solve optimization problems. Included in the Pyomo package are Python classes for defining parameters, variables, and sparse sets, allowing the user to formulate objective functions and constraints. Linear, mixed-integer, non-linear, and non-linear mixed integer models can be formulated in Pyomo for solving large-scale problems. The Pyomo package is part of the Coopr (COmmon Optimization Python Repository) software library (COOPR, 2009). Coopr includes interfaces to common linear, mixed-integer, and nonlinear solvers, allowing users to apply optimizers to models developed using Pyomo. The set of Python packages used by Pyomo are included in a Coopr installation utility.

The event scenarios for this paper were generated by GexCon using their CFD software, FLACS (GexCon, 2011). A set of 279 release scenarios were generated for a real process facility. The process geometry itself is proprietary, however, it represents the full process geometry (equipment, piping, etc.) for a medium-scale facility. All scenarios were considered to have equal probability of occuring, thus $\alpha_a = 1/279$ is the same for all release events $a \in A$. Concentrations were provided at 994 potential point sensor locations and 92 potential line-of-sight sensor locations. Point sensors detect contaminant concentrations in terms of a percentage of the lower flammability limit (LFL). Line-of-sight sensors, on the other hand, detect contaminant concentrations in terms of lower flammability limit meters (LFLm). This provided a total of 1086 potential sensor locations for our placement problem. The measure selected for the damage coefficient in this example was the "time to detect" for each release scenario and each potential sensor location. The time to detection was selected since it was the available output for the set of release scenarios. Of course, other measures, like the volume of the flammable cloud at the time of detection, could instead be used. Data were provided for two sensor detection levels for each sensor type: 10%LFL and 30%LFL for point sensors, 1LFLm and 2LFLm for the line-of-sight sensors. For this optimization problem, we used the lower of the two values for both sensor types (10%LFL and 1LFLm). It should be noted that 9 of the events were not detected by any of the potential sensors at these concentration settings. These events were removed from the data for all results shown here for simplicity. The problem formulations were all written in Pyomo, part of the Coopr 3.0 software library, and solved using CPLEX 12.2 on a dual quad-core Intel(R) Xeon(R) CPU X5570 @ 2.93GHz, with 96 GB RAM and hyperthreading enabled.

Using the first formulation, with $p = 50$, we can solve for an optimal placement in just over 3 seconds using Pyomo and CPLEX on the aforementioned computer. This is very favorable performance given the large number of potential locations. Given the fast solution time, it is computationally efficient to explore a large range for p, the maximum number of sensors, and examine the tradeoff between the number of sensors (resources) and the optimal expected time to detection (performance).

Using the first formulation from the previous sec-

tion, we can solve a sequence of problems with an increasing value for p, the maximum number of sensors. In this formulation, the "dummy" location is added to relax the first constraint and allow for the possibility that a release event is not detected. In this case study, the damage coefficient (or in this problem, "time to detect") for the "dummy" location is set to be one order of magnitude larger than the largest damage coefficient provided by the CFD simulations. This value heavily penalizes any instances where an event is considered not detected, encouraging the detection of all events.

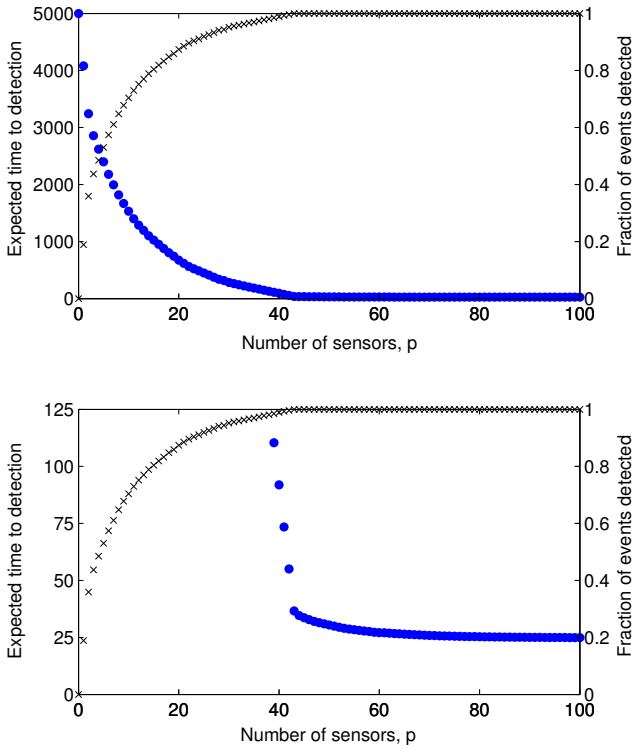

Fig. 1. Expected time to detection (•) and fraction of events detected (×) as a function of the maximum number of allowed sensors, p.

Figure 1 shows the expected time to detect a release with respect to the maximum number of sensors placed. The circular, decreasing data points are plotted along the left axis, which is the expected time to detection as represented by the objective function in equation 4a. The "x" symbol data points, which increase from left to right, correspond to the right hand axis, and represent fraction of events that were detected. The second subplot is simply a zoomed-in view, with the new left axis being scaled from 0 to 120. It can be seen that all of the events are detected when 43 gas detectors are allowed ($p = 43$). This result can be further verified using the second formulation (minimizing the number of sensors required to detect all scenarios) from the previous section. Solving this formulation, we find that the minimum number of sensors required to successfully detect all the included 270 scenarios is 43 total sensors, of which 36 are point sensors and 7 line-of-sight sensors. As mentioned above, in the generation of Figure 1, the large damage coefficient assigned to the "dummy" location encourages detection of all events, so this agreement is not surprising.

While Figure 1 shows the tradeoff between the number of sensors and the expected time to detection, it does not provide any information about the maximum time to detection associated with a particular solution.

To illustrate a robust optimization solution, we can instead choose to optimize using the third formulation, and minimize the maximum time to detection. Results for this formulation as a function of the maximum number of allowed sensors are shown in Figure 2. Since 43 sensors are required to detect all events, the maximum time to detection is infinite for all values of $p < 43$. Therefore, in Figure 2, all p values before 43 are omitted. Above 61 sensors, the optimal objective value does not improve, and the addition of more sensors beyond this number provides no further improvement (for this formulation). This min-max robust optimization formulation is significantly more expensive to solve, although still sufficient for realistically scaled problems. Solving this problem with $p = 50$ required 107 seconds on the aforementioned computer.

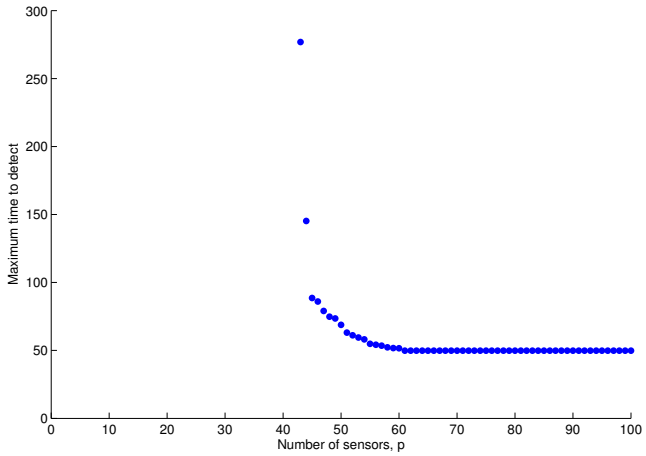

Fig. 2. Maximum time to detect any event for a given number of installed sensors.

Conclusions

In this paper, we adapted a previously developed mixed-integer approach for optimal sensor placement to the problem of placing gas detectors in a petrochemical facility. This required the integration of data from CFD simulation tools with the mixed-integer optimization formulation. Three formulations were developed.

First, the stochastic programming formulation seeks to minimize the expected value of potential damage. The second formulation solves for the minimum number of sensors required to detect all possible events. The third formulation represents a min-max robust optimization formulation and solves for a placement that minimizes the maximum time to detection over all discrete scenarios.

All of these formulations were effective and computationally efficient. Numerical results were presented, including 279 CFD simulations representing different possible leak scenarios, for an actual process geometry. This case study demonstrates that the approach is feasible for gas detector placement with real process facilities. While we have demonstrated the effectiveness of this, there are several improvements that can be made to the base formulation.

In this paper, we used existing gas leak simulations for our multi-scenario formulation as provided by GexCon. The time to solve the optimization formulation is very reasonable, and the time to create these simulations is by far the dominant cost for the overall approach. Given another facility (a new geometry), a single simulation can take hours to days of computational time. Therefore, computing clusters are typically required to generate the necessary scenarios. Furthermore, the characteristic uncertainties (leak location, process conditions, weather, etc.) are continuous variables, and there is no guarantee that enough scenarios were included to effectively approximate this continuous space. Future work will include the development of confidence intervals on the objective function to provide some assurance that a sufficient number of scenarios have been included. PySP, a recent extension to the Pyomo modeling framework, provides techniques to calculate these confidence intervals. This package will be considered in future work. Once confidence intervals are known, additional constraints may be necessary to provide robust solutions when there are only a small number of scenarios. Future work will investigate the robustness of solutions with smaller numbers of scenarios.

Given the large computational time associated with generating scenarios, one would like to extend the value of the scenarios used. Since the computational time associated with the optimal gas detector placement is relatively small, it may be possible to extend this formulation while retaining sufficient solution times. To provide increased solution robustness with the existing scenario set, one could consider a two-stage problem that considers the potential for failure of one of the sensors to detect a particular scenario (Berry et al., 2009). This two-stage problem, which will be significantly larger and more computationally expensive, will be the subject of future work. Additionally, since nuisance trips, or false alarms, are highly undesirable in process facilities, a formulation taking sensor voting into account will be developed.

Acknowledgments

This research was supported in part from the Office of Advanced Scientific Computing Research within the Department of Energy Office of Science as part of the Complex Interconnected Distributed Systems program. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

The authors would also like to thank GexCon for their support in providing data for this paper.

References

Berry, J., Carr, R., Hart, W., Leung, V., Phillips, C., and Watson, J. (2009). Designing contamination warning systems for municipal water networks using imperfect sensors. *Journal of Water Resources Planning and Management*, *135*, 253.

Berry, J., Fleischer, L., Hart, W., Phillips, C., and Watson, J. (2005). Sensor placement in municipal water networks. *Journal of Water Resources Planning and Management*, *131*, 237.

Berry, J., Hart, W., Phillips, C., Uber, J., and Watson, J. (2006). Sensor placement in municipal water networks with temporal integer programming models. *Journal of water resources planning and management*, *132*, 218.

COOPR. (2009). Coopr: A common optimization python repository. http://software.sandia.gov/coopr.

Fire & Safety World Online. (2011). Gas detection. http://www.fs-business.com/-FAQ/FAQGasDetection.asp.

GexCon. (2011). Flacs cfd disperson modeling and explosion software. http://gexcon.com/FLACSoverview.

Hamel, D., Chwastek, M., Farouk, B., Dandekar, K., and Kam, M. (2006). *Measurement Systems for Homeland Security, Contraband Detection and Personal Safety, Proceedings of the 2006 IEEE International Workshop.* . IEEE, 38–42.

Hart, W. E., Watson, J.-P., and Woodruff, D. L. (2011). Pyomo: Modeling and solving mathematical programs in python. *Mathematical Programming Computation*,.

UK Health & Safety Executive. (2011). Fire and gas detection. http://www.hse.gov.uk/-offshore/strategy/fgdetect.htm.