OPTIMALLY LOCATING SURVEILLANCE ASSETS IN URBAN AREAS

Javier Salmeron[(a)], **Kevin Wood**[(b)]

[(a),(b)] Operations Research Department, Naval Postgraduate School, Monterey, CA, 93943, U.S.A.

[(a)]jsalmero@nps.edu, [(b)]kwood@nps.edu

ABSTRACT
This paper develops mathematical-programming models for optimal placement of tower-mounted surveillance systems such as BETSS-C (Base Expeditionary Targeting and Surveillance Systems-Combined). A solution maximizes the "value" that a set of tower-mounted cameras has in covering pre-defined "points of interest" on the ground. Near-optimal solutions for problems with up to 20 towers, 30 candidate locations for those towers and 100 points of interest are produced on laptop computer in under five minutes.

Keywords: camera tower, surveillance, facility location, integer programming, generalized network flow

1. INTRODUCTION

In Iraq and Afghanistan, Coalition Forces have found that camera towers such as "GBOSS" (Ground-Based Operational Surveillance System), BETSS-C (Base Expeditionary Targeting and Surveillance Systems-Combined), and JLENS/RAID PS2 systems can help thwart the emplacement of improvised explosive devices (IEDs). These systems can also identify disturbances to which troops should respond, follow suspicious vehicles, and so on. Their use in populated areas is critical to the security of U.S. and allied military forces as well as local civilian populations. No tool currently exists, however, for assigning a limited number of camera towers to a larger number of potential (secure) sites so as to optimize the "value" of the surveilled "points of interest" (POIs) or to optimize some other appropriate objective.
To address the lack of an appropriate analysis tool, the research described here develops, implements and solves a series of prototypic mathematical models for optimizing camera-tower placement. We create mixed-integer, nonlinear optimization (MINLP) models for this purpose, reformulate those models for tractability, and solve them using general-purpose optimization tools. Results are displayed graphically. We note that the models described in this paper should apply to the optimized placement of aerostats (i.e., tethered, camera-carrying balloons), in conjunction with camera towers or by themselves.

2. MATHEMATICAL MODELS

2.1. A Basic Camera-Tower-Location Model

We first develop a MINLP model for optimizing camera-tower locations. This model, and the others studied in this paper, concern themselves with detecting specific acts at specific points on the ground from individual towers, and not with attempting to identify and track suspicious, moving targets, perhaps across multiple towers. Thus, our models resemble stochastic facility-location models in which a limited set of facilities is opened to serve uncertain customer demands at known locations. Analysts use such models for locating and sizing actual production facilities, but also for locating emergency-services facilities (such as fire stations) and delivery assets (such as ambulances) to meet probabilistically occurring emergencies (such as building fires or medical calls). Snyder (2006) provides a review of such models.
Murray et al. (2006) make explicit use variants on facility-location models in order to locate security monitors effectively. Their bi-objective approach locates cameras (cf. facilities) and accumulates rewards for both single and double coverage of a control point (cf. customer). Their approach does not incorporate compounded probabilities of detection as our models do, however. Hörster and R. Lienhart (2006) address a problem involving both coverage and resolution of images, accounting for cost of operations and effectiveness of a set of orientation-dependent cameras. They have a number of models, but at least one model has a strong flavor of a facility-location model, with variables that determine whether a camera is placed at particular location and with a particular orientation (cf. facility operations), and with rewards that depend on whether a particular control point is covered by (cf. served by) by a camera. Bodor et al. (2007) address the problem of camera placement for maximum observability of moving subjects in a given area, and introduce a joint measure of observability with quality of the view. Their optimization in terms of the "motion statistics of a scene" resembles the optimization of facility locations over an empirical distribution of customer demands. A substantial literature covers more detailed models of camera physics and subject motion but, of necessity, limits the combinatorial aspects of camera placement. For an overview of such models, we refer the reader to Section 3 in Bodor et al. (2007) and the references therein.
Our models are not game-theoretic defender-attacker models (Brown et al. 2006), but is useful to describe them in terms of (a) a *defender* who operates the surveillance system and who will suffer the consequences of undetected attacks, and (b) an *attacker* who attempts to carry out attacks on the defender in a probabilistic fashion. Our first model, **NLPavg1**, follows.

NLPavg1: $\min_{\mathbf{y}} \sum_{i \in I} v_i \prod_{\ell \in L} q_{i\ell}^{y_\ell}$ (1)

subject to:

$$\sum_{\ell \in L} y_\ell \le m \tag{2}$$

$$y_l \in \{0,1\} \forall l \in L, \tag{3}$$

where $\ell \in L$ is a set of potential camera-tower locations; $i \in I$ is a set of POIs that should be kept under surveillance; m is the number of camera towers available, with each having identical capabilities; v_i is the "value" of POI *i*, which represents the damage that the unique "initiating event" (such as an IED emplacement) would cause at i if the event is not detected; $q_{i\ell}$ is the probability of not detecting an event at POI *i* from location ℓ if a tower is placed at that location; and the decision variable $y_\ell = 1$ if a tower is located at ℓ, and $y_\ell = 0$ otherwise. If more than one type of event might occur a POI i (e.g., an IED emplacement or a riot), the POI can be replicated and treated as a separate POI for each event type.

We note that, as described above, each event occurs or does not occur within short timeframe. We maintain that viewpoint for simplicity in descriptions. A surveillance system might be in place for months or years, however, and a POI might suffer from many events over that time. In such a case, the model remains valid, however, if events at i occur according to a Poisson process with known rate (Lin et al. 2013). Now, v_i represents the expected total value of potential attacks on i over the monitoring period if all attacks are successful.

Now, since $\prod_{\ell \in L} q_{i\ell}^{y_\ell}$ is the probability that an event at at POI *i* goes undetected by all of the installed camera towers, **NLPavg1**'s objective, under an assumption of independence, minimizes overall expected value of undetected events across all POIs, subject to the limit on available towers. Henceforth, we use "expected damage" to mean the "expected value of undetected events." In particular, we refer to "expected damage at an individual POI *i*," $v_i \prod_{\ell \in L} q_{i\ell}^{y_\ell}$, and to "overall expected damage," $\sum_{i \in I} v_i \prod_{\ell \in L} q_{i\ell}^{y_\ell}$. We add four notes, also:

(1) **NLPavg1** does assume independence of detections for an event at a given POI, and requires some user inputs that may not be immediately available, namely v_i and $q_{i\ell}$; subjective estimates for these quantities may be required.

(2) The notation hides some of the practical aspects of an implementation. Suppose, for instance, that no line of sight exists between potential camera-tower location ℓ and POI *i*. In this case $q_{i\ell} = 1$ and the model is correct. However, our implementation would not even create the corresponding term in the objective function.

(3) This model and all others in this paper extend in a straightforward fashion to handle various (but fixed) camera configurations at a given location that provide different coverages of an area.

(4) Given that $v_i > 0 \; \forall i$, and given that $\prod_{\ell \in L} q_{i\ell}^{y_\ell}$ is a convex function of continuous y_i, the continuous relaxation of **NLPavg1** is a convex problem. Thus, in theory, **NLPavg1** can be solved using the integer extension of Kelley's cutting-plane algorithm "KCPA"; see Kelley (1960). Our testing of KCPA shows that it performs poorly, however. We have also tested a standard solver that will solve convex MINLPs like **NLPavg1**. Again, computational performance is poor. (Some details will be provided in Section 4.) Because of poor results with "standard methods," we emphasize the conversion of **NLPavg1** as well the next model, into mixed-integer linear programs (MIPs), which can be solved by standard, linear-programming-based branch and bound.

2.2. Minimizing Maximum Expected Damage

A second model, **NLPmx1**, seeks to minimize the maximum expected damage at any POI, i.e., the worst-case damage across all POIs:

NLPmx1: $\min_{\mathbf{y},z} z$ (4)

subject to: (2), (3)

$$z \ge v_i \prod_{\ell \in L} q_{i\ell}^{y_\ell} \qquad \forall i \in I\ , \tag{5}$$

where the new set of constraints ensures that the objective value takes the maximum, across all POIs, of the expected damage at each individual POI. In theory, **NLPmx1** can also be solved via an extension Kelley's cutting-plane algorithm, but a simpler approach exists based on the fact that we can minimize z' = log z without affecting the outcome:

$$\log z \ge \log\left(v_i \prod_{\ell \in L} q_{i\ell}^{y_\ell}\right) \forall i \in I \Rightarrow$$

$$z' \ge \log v_i + \sum_{l \in L} \left(\log q_{il}\right) y_l \;\; \forall i \in I\ . \tag{6}$$

Thus, the following model is equivalent to **NLPmx1**, and can be solved as a MIP:

MIPmx1: $\min_{\mathbf{y},z'} z'$ (7)

subject to:

(2), (3), (6).

NLPmx1 may be a more appropriate model than **NLPavg1** if, roughly speaking, a large number of small-scale attacks spread across a region is deemed less damaging to the defender than a few large-scale attacks that are focused on a smaller area. For example, minimizing overall expected damage seems appropriate when the attacker has limited information about our monitoring methods and our valuations of the various POIs: we expect an adversary or group of adversaries to carry out multiple, somewhat "random" attacks in this case. A worst-case analysis could be more appropriate if the attacker can learn about and selectively attack a few high-value and possibly poorly monitored locations; this relates to defender-attacker models as described by Brown et al. (2006).

Unfortunately, the linearization technique applied to **NLPmx1** does not apply to **NLPavg1**: the logarithm function cannot be used to decompose that model's objective function $\sum_{i\in I} v_i \prod_{\ell \in L} q_{i\ell}^{y_\ell}$ into a linear expression of the *y*-variables. Different linearization techniques apply, however, as described next.

2.3. Converting NLPavg1 into Generalized Network Flow Based Model

This section converts **NLPavg1** into a MIP whose structure may be viewed in terms of generalized network flows (Ahuja et al., 1993, pp. 566-572). Let $L_i = \{\ell \in L \mid q_{i\ell} < 1\}$, let $n(i) = |L_i|$, and assume that L_i is ordered as $L_i = \{\ell_i^1, \ell_i^2, \ldots, \ell_i^k, \ldots \ell_i^{n(i)}\}$. We propose the following model:

NETavg1: $\min_{\mathbf{q}', \mathbf{x}, \bar{\mathbf{x}}, \mathbf{y}} \sum_{i\in I} v_i q_i'$ (8)

subject to:

(2), (3)

$$x_{i,\ell_i^1} + \bar{x}_{i,\ell_i^1} = 1 \qquad \forall i \in I \tag{9}$$

$$x_{i\ell_i^k} + \bar{x}_{i\ell_i^k} = x_{i,\ell_i^{k-1}} + q_{i,\ell_i^{k-1}}\, \bar{x}_{i,\ell_i^{k-1}} \quad \forall i \in I,\ k = 2,\ldots,n(i) \tag{10}$$

$$q_i' = x_{i,\ell_i^{n(i)}} + q_{i,\ell_i^{n(i)}}\, \bar{x}_{i,\ell_i^{n(i)}} \quad \forall i \in I \tag{11}$$

$$0 \le x_{i\ell} \le 1 - y_\ell \qquad \forall i \in I, \ell \in L_i \tag{12}$$

$$0 \le \bar{x}_{i\ell} \le y_\ell \qquad \forall i \in I, \ell \in L_i. \tag{13}$$

For each $i \in I$, the model describes a generalized network flow over a series of paired, parallel arcs. Starting with one unit of flow representing the probability of non-detection of an event at i, the flow first crosses one of two parallel arcs corresponding to $\ell_i^1 \in L_i$. If $y_{\ell_i^1} = 0$, no camera tower is installed at ℓ_i^1, an event at *i* cannot be detected from that location, and the flow traverses the arc corresponding to $x_{i\ell_i^1}$ with no reduction; that is, the probability of non-detection remains one. But, if $y_{\ell_i^1} = 1$, the flow traverses an arc corresponding to $\bar{x}_{i\ell_i^1}$, and the flow received at the end of that arc is reduced to $q_{i\ell_i^1}$; that is, the probability of non-detection of an event at i has been reduced from one to that factor. Repeating this construction for all $\ell_i^k, k = 2,...,n(i),$ means that the flow exiting the last node associated with i and recorded by q_i' equals $\prod_{k=1}^{n(i)} q_{i\ell_i^k}^{y_{\ell_i^k}}$, as required.

2.4. Limited Camera Surveillance

The models **NLPavg1, NETavg1** and **MIPmx1** all assume that if $q_{i\ell} < 1$ and a camera tower is located at ℓ, then a probability of non-detection equaling $q_{i\ell}$ is always achieved from that location. This assumption may be optimistic, because a camera needs time to pan or rotate, tilt, zoom in and out, and focus on each of the POIs assigned to it (Peruzzi 2013). Also, human observers may become less efficient (i.e., probabilities of detection may decrease) if required to monitor too many POIs. Our research has not yet addressed directly these difficult issues, although the work of Burton et al. (2008) may apply. That work determines the proportion of time that a single camera should dedicate to surveilling POI *i*, assuming events of interest occur according to a Poisson process with a location-dependent rate and that detection times at each location are exponentially distributed. (Independence is assumed between POIs.)

We can make our approach more realistic with respect to the issues discussed above, however. To do that, we incorporate a parameter *k* denoting the maximum number of sites that any one camera tower may be assigned to surveil. Additional variables are also defined: $\bar{y}_{i\ell} = 1$ if a tower is located at $\ell \in L$ and is assigned to surveil POI $i \in I$, and $\bar{y}_{i\ell} = 0,$ otherwise. With this new modeling paradigm, **NLPavg1, NETavg1** and **MIPmx1** convert into **NLPavg2, NETavg2** and **MIPmx2**, respectively:

NLPavg2: $\min_{\mathbf{y}, \bar{\mathbf{y}}} \sum_{i\in I} v_i \prod_{\ell \in L} q_{i\ell}^{\bar{y}_{i\ell}}$ (14)

subject to:

(2), (3)

$$\sum_{i\in I} \bar{y}_{i\ell} \le k\, y_\ell \qquad \forall \ell \in L \tag{15}$$

NETavg2: $\min_{\mathbf{q}', \mathbf{x}, \bar{\mathbf{x}}, \mathbf{y}, \bar{\mathbf{y}}} \sum_{i\in I} v_i q_i'$ (16)

subject to:

(2), (3), (9)-(13),(15)

$$0 \le x_{i\ell} \le 1 - \bar{y}_{i\ell} \qquad \forall i \in I, \ell \in L_i \tag{17}$$

$$0 \le \bar{x}_{i\ell} \le \bar{y}_{i\ell} \qquad \forall i \in I, \ell \in L_i\ ; \tag{18}$$

MIPmx2: $\min_{\mathbf{x}, \bar{\mathbf{x}}, \mathbf{y}, \bar{\mathbf{y}}, z'} z'$ (19)

subject to:

(2), (3), (15)

$$z' \ge \log v_i + \sum_{\ell \in L} (\log q_{i\ell}) \bar{y}_{i\ell} \quad \forall i \in I. \tag{20}$$

3. COMPUTATIONAL IMPLEMENTATION

3.1. Optimization Environments

This section tests **NETavg1, MIPmx1, NETavg2** and **MIPmx2** using a number of randomly generated physical settings. All linear models are implemented in Xpress-MP development environment (FICO 2015), and are solved using the Xpress Optimizer, Version 27.01.02. The remainder of the document refers to this implementation, except for brief, specific comments on results obtained using (a) Kelley's cutting plane algorithm on a nonlinear formulation, and (b) one standard algorithm for MINLPs.

The size of the mathematical models varies by scenario (see Section 4). For example, scenario "*Large*9," which applies **NETavg2** on a 30-location, 100-POI example, generates a model with 2,738 variables (899 binary) and 2,739 constraints; scenario "*Large*10," which is identical to "*Large*9," but applies **MIPmx2**, generates a model with 900 variables (899 binary) and 131 constraints. All computational times are for runs performed using a single processor on a Dell Latitude XT2 Core Duo laptop computer, with 5 GB of RAM, and running at 1.60 GHz.

3.2. Database

The supporting database for our tool is implemented in Microsoft Access (Microsoft 2015). Each database file contains one modeling example (corresponding to the "DBQ" input parameter in the Xpress-MP code), which represents an instance of physical layout of POIs and locations. For that instance, the file may include several "scenario" settings that differ, for example, in the number of available cameras or in the type of model to be solved. The structure of this database is as follows (see Figure 1):

Tables: LOC (locations); POI (points of interest); LOC_POI (attributes for locations and points of interest); SCENARIO (different scenarios to run, see Section 4, and associated solutions to store, for the incumbent "example," of locations and POIs).

Queries: Delete_LOC (eliminates all records from LOC table); Delete_POI (eliminates all records from POI table); LOC_POI_CreateMatrix (creates the list of all possible combinations of locations and POIs to ease the input of associated probabilities).

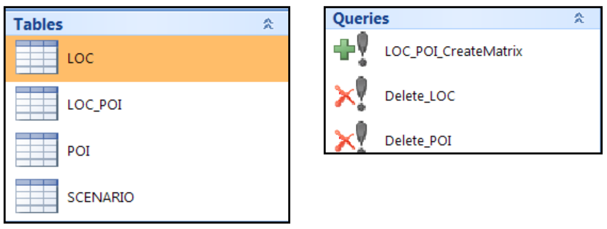


Figure 1. Database tables and queries

Fields in each of the above tables and relationships are shown in Figure 2. Tables 1-4 describe these fields in more detail.

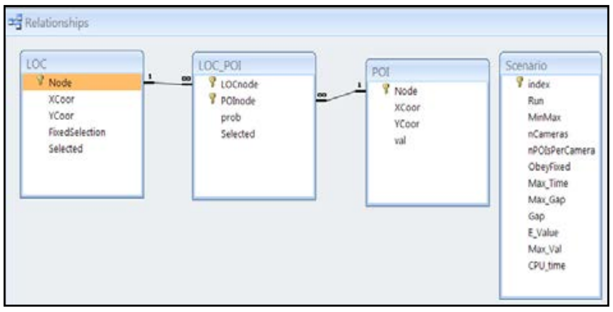


Figure 2. Fields for the database tables, and relationships among tables

Table 1: LOC (Candidate tower locations)

Name	**Type**	**Default**	**Description**
Node	Text		Location code
XCoor	Double	0.0	X coordinate
YCoor	Double	0.0	Y coordinate
FixedSelection	Yes/No	No	Location must be selected?
Selected	Yes/No	No	Location was selected? (OUTPUT)

Table 2: POI (Points of interest)

Name	**Type**	**Default**	**Description**
Node	Text		POI code
XCoor	Double	0.0	X coordinate
YCoor	Double	0.0	Y coordinate
val	Double	1.0	Value of the POI

Table 3: LOC_POI (Attributes by LOC and POI)

Name	**Type**	**Default**	**Description**
LOCnode	Text		Location code
POInode	Text		POI code
prob	Double	0.0	Probability of detection at the POI from the location
Selected	Yes/No	No	POI selected to be surveilled from the location? (OUTPUT)

Table 4: SCENARIO (Parameters, options, etc.)

Name	Type	Default	Description
Index	Long Integer		Scenario index (AUTOMATED)
Run	Yes/No	Yes	Run this scenario?
MinMax	Yes/No	No	Solve **MIPmx** (Yes) or **NETavg** (No). (Variants **1** or **2** depending on the number of POIs per camera)
nCameras	Long Integer	0	Number of camera towers allowed
nPOIsPerCamera	Double	2	Number of POIs each camera tower may surveil at a time. Enter 0 if unlimited.
ObeyFixed	Yes/No	No	Obey all fixed selections specified in LOC table?
Max_Time	Long Integer	100	Maximum run time (seconds)?
Max_Gap	Double	0.0	Maximum optimality gap?
Gap	Double		Actual gap? (OUTPUT)
E_Value	Double		Overall expected damage? (OUTPUT)
Max_Val	Double		Maximum single damage? (OUTPUT)
CPU_time	Double		Computational time (seconds)? (OUTPUT)

3.3. Graphical Input and Output Environment

Xpress-MP's embedded graphical displays help visualize the problem and its solution. For example, Figure 3 shows a snapshot mapping out POIs and candidate camera-tower locations; the values for POIs are displayed, also. By clicking on the "Visible" toggle, we would see a series of lines connecting candidate locations with those POIs that could be surveilled, with strictly positive probability of detection.

After the model is run, the "Selected" toggle turns on the display of the following: (a) optimized tower locations, (b) the type of model solved, i.e., average ("avg") or min-max ("mx"), and (c) the number of camera towers available. "Sel. Visible" (Selected Visible) toggles a display that shows which camera towers are assigned to which POIs.

4. COMPUTATIONAL RESULTS

This section presents results for two hypothetical examples, "Small Example" and "Large Example." Each example has a specific "physical setting," which connotes geographical data on candidate locations and POIs, POI values, and probabilities of event detection by POI and location.

An example may also have several parametric variants called "scenarios." A scenario includes the original physical setting from the example, but adds certain parameter values and chooses which optimization model to apply. For instance, we can use the geographical layout of Small Example and create one scenario that allows more camera towers than another, or that seeks to optimize **NETavg1** rather than, say, **MIPmx1**. The scenario is completed by filling in the data for the scenario record (for example, see, Figure 5 in Section 4.1). A user can create a rich variety of scenarios for the same example by just changing a few input parameters and/or toggle settings as identified in Table 4. For example, the user can set the number of camera towers available, toggle the use an ``mx model'' or an "avg model." and specify solution-algorithm parameters (e.g., maximum run time allowed).

Unless otherwise noted, all the scenarios are set to run until a 0% optimality gap is achieved, or a maximum time limit of 300 seconds is reached. No locations are preselected to receive a camera tower.

4.1. Small Example

The physical layout in this example (Figure 3) has ten potential camera-tower locations and eight POIs. Figure 4 enlarges a portion of the example with visibility links activated and associated probabilities of detection displayed. For example, the probability of detecting POI I4 from location L8 is 0.707.

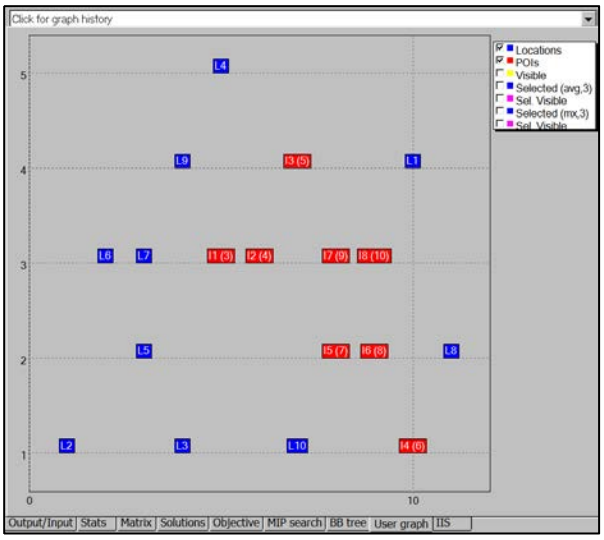


Figure 3. Preliminary display of locations (blue) and POIs (red)

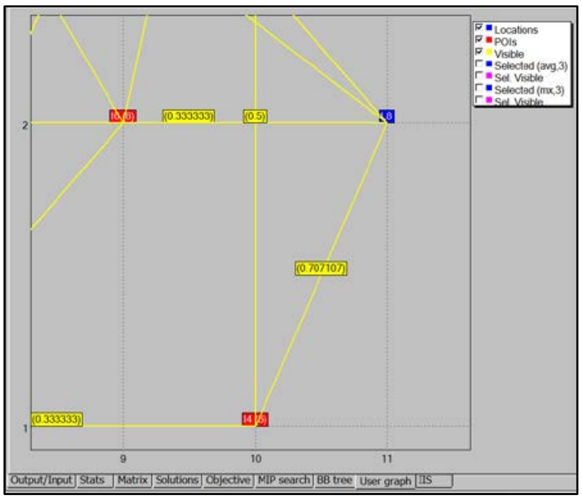
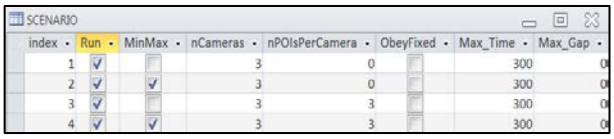


Figure 4. A portion of Small Example enlarged to show lines of sight, two POIs and one camera-tower location.

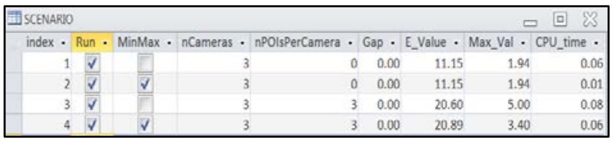
We run four scenarios for this example, as indicated in Figure 5. (From here on, we use *Small*n to refer to the n-th scenario for Small Example, where the index n is automatically produced by the database program.) *Small*1, as modeled and solved, seeks to minimize overall expected damage by applying **NETavg1**. Each of three available camera towers can surveil an unlimited number of POIs simultaneously (indicated with a default value of zero in the data). *Small*2 is identical to *Small*1, but a different model, **MIPmx1,** applies; that is, we seek to minimize the maximum damage at any individual POI. *Small*3 and *Small*4 are identical to *Small*1 and *Small*2, respectively, except that they limit the number of POIs that can be surveilled from any one location to a maximum of three. Accordingly, we apply **NETavg2** to solve *Small*3 and **MIPmx2** to solve *Small*4.


SCENARIO

index	Run	MinMax	nCameras	nPOIsPerCamera	ObeyFixed	Max_Time	Max_Gap
1	✓	☐	3	0	☐	300	0
2	✓	✓	3	0	☐	300	0
3	✓	☐	3	3	☐	300	0
4	✓	✓	3	3	☐	300	0

Figure 5. Small Example scenarios (*Small*1,...,*Small*4)

Figure 6 summarizes results for the Small Example scenarios. *Small*1 and *Small*2 produce similar solutions: the optimal *Small*1 objective (for **NETavg1**) yields an expected damage, over all POIs, of 11.15; see “E_Value” output. Here, the largest, expected damage for a single POI is 1.94, as seen under “Max_Val.” In fact, this is the minimum Max_Val achievable, as shown when model **MIPmx1** is applied in *Small*2. By coincidence, the converse occurs in this example: the E_Value in the *Small*2 solution matches the minimum E_Value obtained for *Small*1. (This coincidence seems unlikely, in general, because instances of **MIPmx1** may have many optimal solutions.)


SCENARIO

index	Run	MinMax	nCameras	nPOIsPerCamera	Gap	E_Value	Max_Val	CPU_time
1	✓	☐	3	0	0.00	11.15	1.94	0.06
2	✓	✓	3	0	0.00	11.15	1.94	0.01
3	✓	☐	3	3	0.00	20.60	5.00	0.08
4	✓	✓	3	3	0.00	20.89	3.40	0.06

Figure 6. Results for Small Example scenarios

Scenarios *Small*3 and *Small*4 are restrictions of *Small*1 and *Small*2, respectively. E_Value for *Small*3 increase to 20.60 from *Small*1’s value of 11.15, and “Max-Value” increases for *Small*4 to 3.40 from Small2’s value of 1.94. Figure 8 displays the solutions. We observe that **NETavg2**’s solution leaves two POIs without any surveillance in *Small*3, and one of those unsurveilled POIs (I3) defines the maximum expected damage (Max_Val equals 5.0). On the other hand, when Max_Val is minimized using **MIPmx2** in *Small*4, the largest, expected damage occurs at another POI (I7, with Max_Val equaling 3.4).

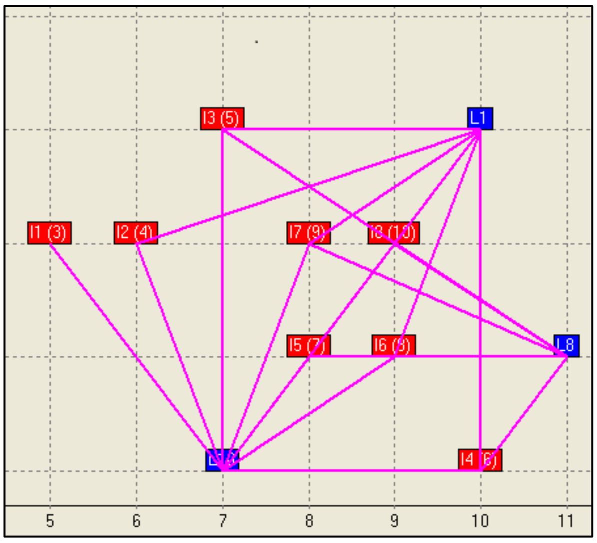


Figure 7. Graphical solution to both *Small*1 and *Small*2 scenarios.

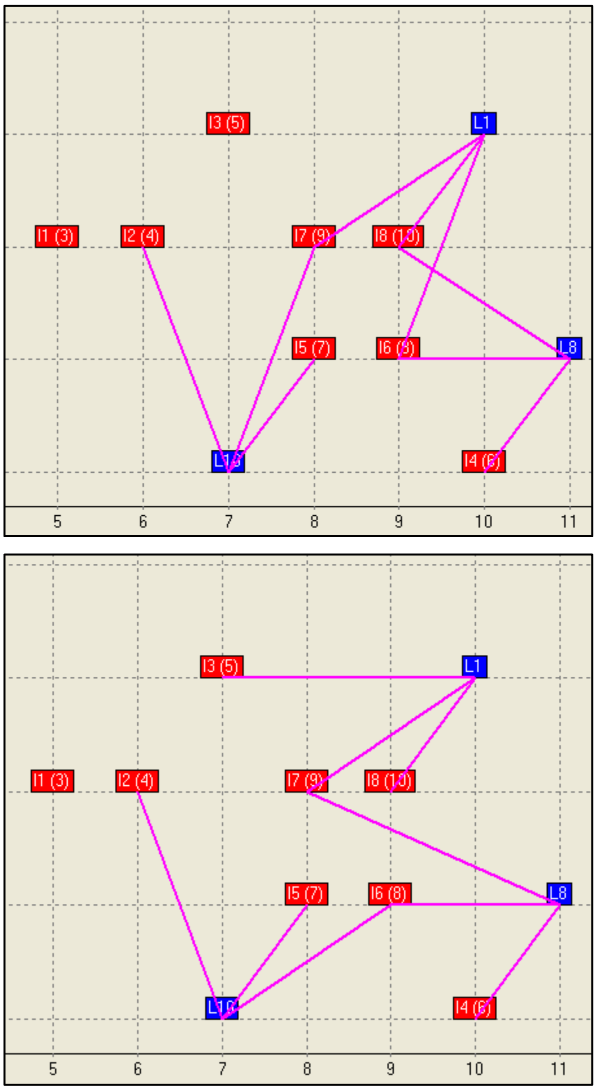


Figure 8. Graphical solutions to *Small*4 (top) and *Small*4 (bottom) scenarios

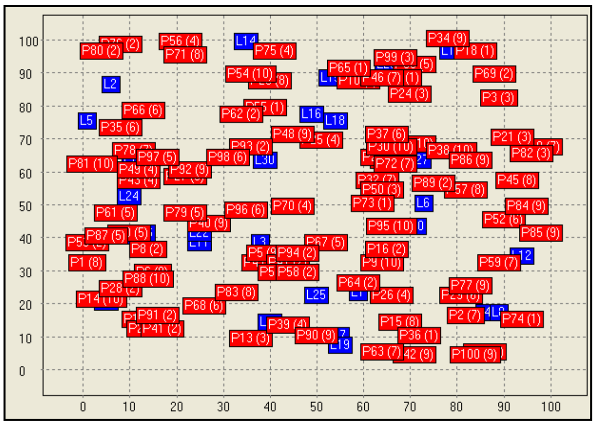
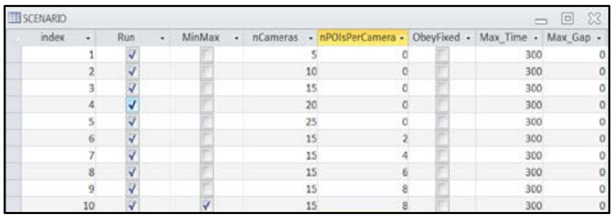


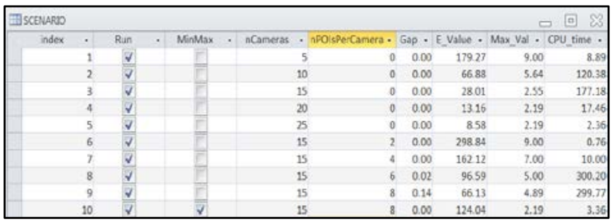
Figure 9. Large Example with 30 locations and 100 POIs



SCENARIO

index	Run	MinMax	nCameras	nPOIsPerCamera	ObeyFixed	Max_Time	Max_Gap
1	✓		5	0		300	0
2	✓		10	0		300	0
3	✓		15	0		300	0
4	✓		20	0		300	0
5	✓		25	0		300	0
6	✓		15	2		300	0
7	✓		15	4		300	0
8	✓		15	6		300	0
9	✓		15	8		300	0
10	✓	✓	15	8		300	0

Figure 10. Large Example scenarios (*Large*1,..., *Large*10)



SCENARIO

index	Run	MinMax	nCameras	nPOIsPerCamera	Gap	E_Value	Max_Val	CPU_time
1	✓		5	0	0.00	179.27	9.00	8.89
2	✓		10	0	0.00	66.88	5.64	120.38
3	✓		15	0	0.00	28.01	2.55	177.18
4	✓		20	0	0.00	13.16	2.19	17.46
5	✓		25	0	0.00	8.58	2.19	2.36
6	✓		15	2	0.00	298.84	9.00	0.76
7	✓		15	4	0.00	162.12	7.00	10.00
8	✓		15	6	0.02	96.59	5.00	300.20
9	✓		15	8	0.14	66.13	4.89	299.77
10	✓	✓	15	8	0.00	124.04	2.19	3.36

Figure 11. Results for Large Example scenarios

4.2. Large Example

This example has 100 POIs to be surveilled from some subset of 30 candidate camera-tower locations (Figure 9). We run ten scenarios, *Large*1,...,*Large*10 (see Figure 10): *Large*1-*Large*5 use **NETavg1** to allocate 5, 10, 15, 20 or 25 towers, respectively, with unlimited surveillance for each tower; *Large*6-*Large*9 fix the number of available camera towers to 15, and solve **NETavg2** with per-tower surveillance limits of 2, 4, 6 and 8 POIs, respectively; *Large*10 solves the 15-tower, 8-POIs-per-tower problem using **MIPmx2**.

Figure 11 displays results. We note, for example, that all unlimited-surveillance scenarios solve optimally in the allotted time. This is not the case for *Large*8 and *Large*9 limited-surveillance scenarios, where 2% and 14% optimality gaps remain after 300 seconds of computation.

On the other hand, *Large*10 solves quickly. Recall that *Large*10 is identical to *Large*9, except that *Large*10 minimizes the largest expected damage for a single POI (Max_Value), while *Large*9 minimizes overall expected damage (E_Value). Outcomes are notably different for *Large*9 and *Large*10. In particular, Max_Value is over 100% greater (worse) for *Large*9 than for *Large*10 and, conversely, E_Value for *Large*10 is almost 100% greater (worse) than for *Large*9.

Figure 12 graphically depicts the solutions for the two scenarios. (POI names are hidden in the displays for the sake of clarity.) We observe that, for the most part, the scenario solutions place camera towers at different locations. But, when a location such as L27 at coordinates (72, 61) is selected under both scenarios, the surveilled POIs are different.

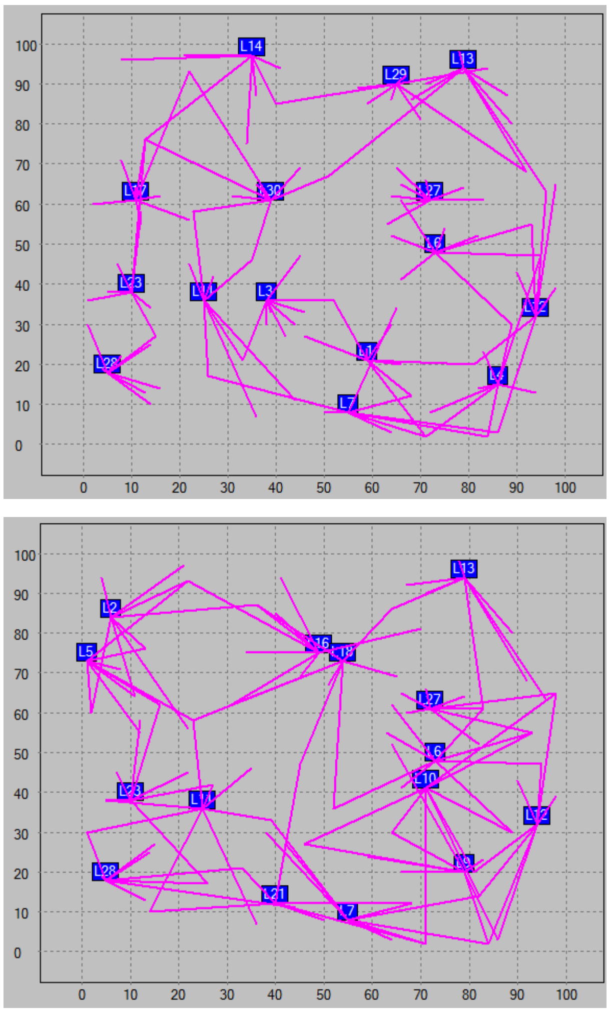


Figure 12. Graphical solution displays for scenarios *Large*9 (top) and *Large*10 (bottom).

As discussed in Section 2.1, it is possible, in theory, to solve **NLPavg1** and **NLPavg2** using (a) variants of KCPA (Kelley 1960) and (b) a standard MINLP solver. We have implemented (a) and (b) for our Small- and Large-Example scenarios using the GAMS algebraic modeling system (McCarl et al. 2014). Specifically, we use CPLEX 12.4 (GAMS 2015, pp. 109-160) to solve master problems in our own implementation of KCPA, and we use DICOPT (GAMS 2015, pp. 189-208) as a general MINLP solver; our implementation of DICOPT employs CPLEX 12.4 for solving MIP master problems

and MINOS (GAMS 2015, pp. 323-354) to solve continuous, non-linear subproblems.
For Small-Example scenarios, DICOPT and KCPA produce optimal solutions in times that are comparable to, or only modestly longer than, those reported in Figure 6. On the other hand, with a few exceptions, neither DICOPT nor KCPA solve Large-Example scenarios efficiently. For example, DICOPT solves *Large1*, which is the smallest of the Large-Example scenarios, in only 2 seconds, but it produces a suboptimal solution having an E_Value of 203.27, rather than an optimal solution, which has an optimal E_Value of 179.27. Relative optimality gaps become even worse as the complexity of the scenarios increases. For example, *Large*9 results in an E_Value equaling 138.30, yet the optimal value is 66.13. Finally, we note that KCPA converges to the optimal solution of the scenarios mentioned above, but even the smallest scenario takes hundreds of iterations to solve and requires computation time that exceeds 1,900 seconds.

5. CONCLUSIONS AND FUTURE RESEARCH

Our work should be extended to more accurately assess and incorporate the "information value" of a collection of POIs that might be assigned to one or more camera towers for surveillance. Exactly how to carry this out is unclear, but we see three key issues:
(a) The current implementation assumes a simple additive or separable value function that ignores "scheduling issues." But, a camera that is set to surveil a collection of POIs may be programmed to focus on, zoom in on, and surveil each POI for a given amount of time before transitioning to another POI. The corresponding surveillance and transition times affect the value of the information collected (for example, the probability that an IED emplacement is detected), and should be part of the optimization process.
(b) Our current models assume constant conditions, but the time of day and weather can affect probabilities of event detection. Naturally, this variability could influence optimal camera-tower placements.
(c) We ignore the possibility that mobile surveillance systems such as UAVs may operate in conjunction with camera towers.

ACKNOWLEDGMENTS
We thank the Joint Improvised Explosive Device Defeat Organization for support that inspired this work.

REFERENCES

Ahuja R.K., Magnanti T.L., Orlin J.B., 1993. Network Flows. Theory, Algorithms and Applications. Upper Saddle River, NJ:Prentice Hall.
Bodor, R., Drenner, A., Schrater, P., Papanikolopoulos, N., 2007. Journal of Intelligent and Robotic Systems 50:257–295.
Burton D., Kress M., Lin K., Rowe A., Szechtman, R. 2008. Optimal Mix and Employment of Sensors for Persistent Surveillance – Final Report. Project Report (obtained from authors at the Naval Postgraduate School, Monterey, CA, U.S.A).
Brown G., Carlyle, M. Salmerón J., Wood K., 2006. Defending critical infrastructure. Interfaces 36: 530-544.
FICO, 2015. FICO Xpress Optimization Suite. Available from: http://www.fico.com/en/products/fico-xpress-optimization-suite.
GAMS, 2015. GAMS—The solver manuals. Washington, DC: GAMS Development Corporation.
Hörster, E. and Lienhart, R., 2006. On the Optimal Placement of Multiple Visual Sensors. Universität Augsburg, Institut für Informatik, Report 2006-18.
Kelley J.E., 1960. The cutting-plane method for solving convex programs. Journal of the Society for Industrial and Applied Mathematics 8:703-712.
Lin K.Y., Atkinson M.P., Chung, T.H., Glazebrook K.D., 2013.. A graph patrol problem with random attack times. Operations Research 61: 694–710.
McCarl B.A., Meeraus, A., van der Eijk P., Bussieck M., Dirkse S., Steacy P., Nelissen F., 2014. McCarl GAMS user guide. Washington, DC: GAMS Development Corporation.
Microsoft, 2015. Microsoft Access 2013. Available from: https://products.office.com/en-us/access.
Murray A.T., Kim K., Davis J.W., Machiraju R., Parent R., 2007. Coverage optimization to support security monitoring. Computers, Environment and Urban Systems 31:133-147.
Peruzzi L., 2013. Borders and base perimeters: No trespassing please! Armada International, Issue 1:4-6.
Snyder L.V., 2006, Facility location under uncertainty: a review. IIE Transactions 38:547-564.