optimiSing THE BARRIER coverage of A Wireless Sensor Network with hub-and-spoke topology using MATHEMATICAL and Simulation models

|  |  |
| --- | --- |
| Mumtaz Karatas | Bhakti Stephan Onggo |
| 1. Department of Industrial Engineering | Centre for Operational Research, Management |
| Naval Academy, National Defense University | Sciences and Information Systems |
| Istanbul, 34940 | Southampton Business School |
| Turkey | University of Southampton, UK |
|  |  |
| 1. Department of Industrial Engineering |  |
| Bahcesehir University |  |
| Istanbul, 34353  Turkey |  |

*Corresponding author: Mumtaz Karatas, email: mkaratas@dho.edu.tr*

ABSTRACT

Deploying wireless sensor networks (WSNs) to form a barrier that provides surveillance against illegal intruders (or targets) is a fundamental sensor-allocation problem. In this paper, we consider a WSN barrier coverage problem with hub-and-spoke topology in which communication hubs form central hubs that are connected to sensors via spokes. We develop and compare three models. The first two are mathematical models: an Integer Non-Linear Program (INLP) and an Integer Linear Program (ILP). The third is an Optimisation-via-Simulation (OvS) model, which comprises agent-based simulation and heuristics. We consider the following factors in the models: multiple sensor and target types, probabilistic detection function, sensor reliability, communication range, communication interference, network topology and budget constraints. The experiment shows that the INLP solutions are close to ILP global optimum solutions. The ILP model is only practical for a small problem, while the INLP and OvS models can cope with bigger ones. OvS can handle more realistic assumptions more easily, such as intelligent targets that can move freely across the barrier, and the movement speed depends on the path taken. To solve this using ILP, we need to reformulate the ILP model but it is very challenging. Hence, our contributions are twofold: (1) our models are more elaborate than other models in the literature and (2) this is the first work that demonstrates how OvS is used to solve a barrier coverage problem and demonstrates its benefit of handling more realistic assumptions.

**Keywords:** Coverage; wireless sensor networks; location; simulation optimisation; agent-based simulation.

# INTRODUCTION

A *Wireless Sensor Network* (WSN) is a wireless network consisting of spatially distributed autonomous sensors to monitor physical or environmental conditions. The relevance of WSN to our lives has become more significant with a wide range of applications, such as supply chain, smart factory, smart home, security and healthcare. In this paper, we focus on applications that require WSN to monitor and watch over a certain area or infrastructure. An issue involving these WSN applications is known as the WSN coverage problem. The coverage of a sensor is defined as the geographic region where the sensor can sense (detect) targets. The coverage function and the distance between the sensor and the target are the inputs used to determine the coverage measure of that particular target. In other words, a target (or a space point) is said to be covered if its coverage measure satisfies some predefined threshold (Wang 2010).

There are three types of WSN coverage problem: area coverage, point coverage and barrier coverage. This paper focuses on the barrier-coverage problem. The objective of a barrier-coverage problem is to prevent a target penetrating a belt region or a one-dimensional line segment that separates one region from another. Examples of barrier-coverage problems include protecting territory borders against intruders (e.g. illegal immigrants, terrorists, smugglers) and protecting extended objects (e.g. oil pipelines, communication lines) against attackers.

In a *barrier-coverage problem*, decision-makers mostly seek to determine sensor-deployment strategies which maximize coverage or guarantee achieving a certain level of coverage. The optimum deployment strategy is affected by operational factors (e.g. deployment budget, limited accessibility to the field and limited operation time) and technical factors (e.g. sensitivity, specificity, reliability, energy consumption and communication range). These factors make determining the optimal allocation of different types of sensors in a WSN barrier against different types of targets difficult. The difficulty is further increased when the sensing region of a sensor is not simply defined by a definite range, but via a probabilistic detection function where the detection probability is a function of the distance between a target and a sensor.

The primary objective of this paper is to find an optimum deployment strategy for multiple types of sensors to detect multiple types of targets. Our solutions take into account sensor characteristics (sensitivity, specificity, reliability and communication range), communication-hub characteristics (interference, range, topology), budget and probabilistic detection function. To the best of our knowledge, there is no existing study that deals with the optimisation and performance measurement of a WSN barrier and considers these factors in a single model. We propose an *Integer Non-Linear Programming* (INLP) model to solve the problem. Next, using a special linearization technique, we approximate the INLP model with an *Integer Linear Programming* (ILP) model. The secondary objective of this paper is to propose an agent-based simulation model to estimate the performance of a deployment strategy against more realistic assumptions. The agent-based simulation (ABS) model and a heuristic optimisation algorithm form our *Optimisation-via-Simulation* (OvS) model. The literature shows that OvS has not been applied to solve barrier coverage problem. This is unfortunate since the use of ABS model allows us to model changes in the behaviour of targets over time. In this paper, we demonstrate how an ABS model can be used as part of OvS to optimise barrier coverage when targets are intelligent (capable of determining weak zones in the barrier) and there are infinitely many path alternatives across the barrier.

The remainder of this paper is organised as follows. We review the literature on the WSN barrier-coverage problem in Section 2. It shows the lack of studies that consider multiple types of sensors and targets, sensor characteristics, communication-hub structures, budget and probabilistic sensing in a single model. It also shows that OvS has not been applied to the WSN barrier-coverage problem. Section 3 explains the terminology and assumptions used in this paper. This section is followed by a detailed description of our INLP, ILP and OvS models in Section 4. Section 5 shows the validation of our models and the results of our experiments. The validation is done by cross-validating the three models (INLP, ILP and OvS). The first experiment compares the three models and assumes that targets do not change their behaviour over time. The main objective of this experiment is to evaluate how close the INLP and OvS results are to the ILP results and to compare their execution times. Afterwards, in a second experiment, we demonstrate the benefit if using OvS by relaxing the assumptions so that the targets can learn about the probability of detection along the barrier by communicating with targets that have been successful in penetrating the barrier and considering news about targets that have been detected (and make news). Furthermore, the targets’ movement speeds depend on the paths taken by the targets. Finally, we conclude our paper in Section 6.

# RELATED WORK

## Barrier Problem

First introduced by Gage (1992), the barrier-coverage problem mostly deals with finding an effective deployment scheme of sensors to detect targets crossing a barrier line. Benkoczi et al. (2015) provide a more recent example that considers a barrier as a line segment and develop a model to solve the problem of determining the locations of mobile sensors with arbitrary coverage ranges. Liu and Towsley (2004) extend the barrier-line problem to a two-dimensional problem by treating a barrier as a belt-shaped region.

The literature shows the dominance of the use of analytic methods and mathematical models to tackle the barrier-coverage problem. Galyaev and Maslov (2011) and Washburn (2015) used game theory to analyse barrier problems for mobile sensors. Galyaev and Maslov (2011) determined optimal patrol trajectories and speeds for sensors performing barrier searches in a water environment. Representing the target detection probability as a nonlinear function of target and searcher speeds, the authors employed game theory to solve the problem. Washburn (2015) considered straight-line and circular barriers with mobile sensors and developed an analytic theory based on a model using a two-person zero-sum game to analyze penetration conditions. In his earlier work, Washburn (1982) derived an upper bound on the probability of detecting a target with a mobile sensor that is patrolling a channel. He also discussed possible patrol paths for which the probability of detection is quite close to the upper bound.

The difficulty of developing an effective barrier depends on factors such as the environment where the patrolling is performed (i.e. air, water or ground), sensor types (mobile, stationary) and their capabilities (detection range, reliability etc.), assumptions about the environment and targets, and the amount and type of information about the operation. Arora et al. (2004) provide a detailed review of the performance of WSNs applied to intrusion detection. They analyse the impact of several factors, such as barrier design, sensor unreliability, environment, communication constraints, target types, reliability and energy consumption, on the performance of a barrier. Cardei and Wu (2006) and Fan and Jin (2010) also provide a good review of the key factors and issues surrounding barrier coverage in a WSN. Researchers have been trying to include as many key factors as possible in one model to make their models more realistic, as shown in the following examples.

Chen et al. (2007) considered developing a barrier along a belt-shaped region using sensors of the same type. The authors proposed a local barrier-coverage model which ensured that no target could penetrate the barrier without being detected. Chen et al. (2008) proposed a methodology to determine and repair weak sensing zones of a barrier to address the reliability issue. He et al. (2012) focused on budget constraints in their model. In subsequent work, they included energy consumption in their model (He et al. 2013, 2014). Similarly, Cheng et al. (2014) considered the energy-management problem of sensor networks and proposed a density-barrier construction algorithm which minimizes the moving distances of mobile sensors. Saipulla et al. (2008, 2009) studied the performance of air-dropped sensor barriers under the assumption that sensor locations are affected by environmental factors, such as wind and terrain. Then, they analysed the conditions to develop a strong barrier. In another study, Saipulla et al. (2013) considered the performance of barriers for different sensor-deployment strategies and studied the impact of sensor mobility on coverage. Ssu et al. (2009) studied the performance of barriers composed of randomly deployed sensors with directional sensing capability. They modelled their barrier as a belt-shaped region and considered energy and sensing-range constraints.

Different from previous studies, Chen et al. (2015), Gong et al. (2016) and Wang et al. (2016) consider barrier-coverage problems for bistatic sensor networks which consist of two types of sensors, transmitters and receivers, working in pairs. In bistatic barriers, the relative position and separation distance of each transmitter/receiver pair affect the coverage performance of that pair significantly, hence they must be planned carefully.

Wang et al. (2017a) considered a barrier-coverage problem for sensors with location errors. They propose a fault-tolerant barrier-coverage formation algorithm which determines the optimal locations of mobile sensors. Wang et al. (2017b) analysed the use of probabilistic coverage models and uncertainties in WSN coverage problems. Karatas and Onggo (2016) considered a simple case of a barrier coverage problem in which multiple types of sensors are located to candidate locations along a line-shaped barrier with the objective of maximizing the total detection probability of targets crossing it. The authors formulated an INLP model to solve the problem. In a more recent study, Karatas (2018) tackled a specific sensor location problem which considers locating a given number of sensors along a belt-shaped region for a hybrid barrier and point coverage application. In particular, the author developed optimisation models and a genetic algorithm (GA) for locating sensors with the objective of protecting critical facilities inside the region (point coverage application) and detecting targets attempting to penetrate the region (barrier coverage application). He included factors such as multiple sensor and target types, and cooperative coverage, in his model. However, he did not incorporate budget, topology, communication, and interference constraints as well as issues such as sensor costs, hub capacities, intelligent targets, etc. Interested readers can also refer to Karatas et al. (2019) for a review of location problems, including barrier coverage problems, observed in military context.

Our literature review pertaining to barrier-problem applications in sensor networks shows the dominance of mathematical models and that the factors and extensions we consider in this paper (i.e. multiple sensor types, target types, probabilistic detection function, unreliable sensors, topology and communication requirements, interference and budget constraints) have been discussed individually but not in a single model. Although there are studies that have attempted to include a number of different factors, e.g. mobile sensors and energy constraints, in their models, to the best of our knowledge, no one has considered a generalized barrier problem setup as defined in this paper.

## Optimisation via Simulation

An Optimisation via Simulation (OvS) model is an optimisation model in which the objective function can only be estimated by running a simulation model (in contrast to the objective function in an ILP or INLP model that can be computed exactly) subject to a set of constraints that can be defined explicitly (as in ILP or INLP models or implicitly embedded in a simulation model). Hence, an OvS algorithm typically iterates between a step that explores possible solutions and a step that estimates the objective function until a certain stopping condition is reached. This is different from the use of a simulation to validate or test the robustness of a solution obtained from an optimisation model (e.g. Wang et al. 2009, Tao et al. 2012, Karatas and Onggo 2016).

OvS has been studied for decades. One of the earliest reviews of OvS research was published in 1987 (Meketon, 1987). The author summarized the early development of OvS techniques, such as response surface and stochastic approximation. Recently, Fu (2015) provided a comprehensive overview of OvS models. He classifies OvS based on whether the decision variables are continuous, discrete (ordered or unordered, finite or infinite) or a combination of the two. Each category can be solved using specific techniques (e.g. Ranking-and-Selection is suitable for finite unordered decision variables and response-surface methodology is suitable for continuous decision variables). The same classification is also used in Abo-Hamad and Arisha (2011). They also classify the techniques into several groups, i.e. gradient-based, meta-model-based, statistical and random search/meta-heuristics. The combination of metaheuristics and simulation for discrete optimisation problems is referred to as simheuristics and reviewed in Juan et al. (2015). Amaran et al. (2016) classify OvS techniques based on whether they are applicable to problems with discrete or continuous variables, and whether they apply local or global optimisation. Xu et al. (2015) classify OvS techniques based on how they handle the noisy, computationally expensive, and black-box nature of simulation optimization.

OvS has been applied in several areas (Amaran et al. 2016, Xu et al. 2015). One of them is healthcare. Decision-making in healthcare needs to deal with uncertain factors, such as the arrival of emergency patients and the time needed to treat patients. For example, Ahmed and Alkhamis (2009) used a statistical technique to find the optimum number of resources (doctors, nurses and lab technicians) in an emergency department to maximize patient throughput and reduce patient time in the system, subject to deterministic constraints (i.e. total budget and bounds on the number of resources) and a stochastic constraint (i.e. average waiting time). Their model belongs to discrete OvS with a stochastic constraint.

Scheduling is another area that often deals with uncertainty. OvS has been applied to generate optimal staff scheduling in call centres. This is because call-centre operations need to handle stochastic call arrivals and the variability of the time needed to serve callers. For example, Avramidis et al. (2010) combined simulation with ILP to determine an optimum staff schedule in a call centre that minimized the total cost of staff subject to several expected service-level constraints. Scheduling decisions in manufacturing also need to deal with stochastic demand and supply. Lin and Chen (2015) combined GA and optimal computing budget allocation (OCBA) approach to generate an optimal flow shop schedule in a semiconductor assembly facility.

Routing and inventory problem is another example where stochasticity is inherent (e.g. travel time, lead time, demand). De Keizer et al. (2015) used a hybrid Mixed Integer Linear Programming (MILP) and discrete-event simulation model to address the design of logistics network to distribute perishable products. Juan et al. (2014) applied simheuristics method to minimize the total inventory and routing cost with stochastic demand. Transportation is an important green gas emission source, especially in cold supply chains because products must be stored and transported at low temperatures. Saif and Elhedhli (2016) combined Mixed Integer Programming and discrete-event simulation model to minimize the total cost of a cold supply chain which took into account routing, inventory and green gas emission costs. GA metaheuristics was used in McCormack and Cotes (2015) to find good base-station locations and determine emergency-vehicle fleet allocation at each station to maximize the expected survival probability of patients.

Nature is inherently stochastic. This include renewable energy sources (e.g. wind) and natural disaster (e.g. wildfire). Hence, OvS is one of the techniques that has been applied to help decision-making that is related to nature. For example, Yin et al. (2017) used GA metaheuristics to determine optimum wind-farm micro-siting which considers wind uncertainty (e.g. direction, speed and probability of occurrence). A Monte Carlo simulation model was used to estimate energy production. Chang and Lin (2015) used a meta-model-based technique to establish an optimum hybrid renewable-energy system design that minimized the expected total cost subject to meeting the demand for power. They used a Monte Carlo simulation to estimate the total cost. Rytwinski and Crowe (2010) developed a cellular automata simulation to model the stochastic and complex behaviour of wildfire. The optimisation part was implemented using OptQuest® which used a metaheuristic technique. The combined simulation optimisation model was used to find optimal fuel-break locations that could minimize fire risk.

OvS has also been applied in maintenance management and supply-chain management as both deal with uncertainty. The application in these two areas have been reviewed by Alrabghi and Tiwari (2015) and Abo-Hamad and Arisha (2011), respectively. OvS has also been applied to estimate the parameters of their simulation model (Kuo et al. 2016).

The review in this section is not meant to be exhaustive but to give a few examples in which OvS have been applied in different areas. There are many literature review papers on OvS. Amaran et al. (2016) provide a good summary of OvS literature reviews published until 2011.

The above examples show that OvS is suitable for an optimisation problem that tackles a complex stochastic system where the objective function must be estimated using simulation. The barrier-coverage problem belongs to the category of complex stochastic systems. However, to the best of our knowledge, OvS has not been used to solve the barrier-coverage problem. It should be noted that, in the barrier-coverage problem, simulation has been widely used to test the correctness or robustness of optimisation-model results (e.g. Wang et al. 2009, Tao et al. 2012, Karatas and Onggo 2016). In this case, simulation is used after the optimisation process finishes, which is different from OvS where simulation is used to estimate the objective function during the optimisation process.

# PROBLEM STATEMENT and model formulation

In this paper, we assume that the barrier consists of a belt region of length *L* and width *W*. The communication hubs and sensors, which form the barrier, are deployed using hub-and-spoke topology. Many types of networks, including WSNs, often adopt hub-and-spoke topology (also called star topology) to improve the performance of the system by efficiently routing flows between origins and destinations. In a hub-and-spoke network, hubs work as connecting central nodes between specified destinations called “spokes”. Today’s expensive WSN infrastructures mostly depend on centrally-deployed hub-and-spoke networks (Basagni et al. 2004). Lying at the heart of the network-design domain, hub-and-spoke network applications are abundant in fields such as the military, telecommunications, computer networks and transportation (Contreras 2015). Farahani et al. (2013) provide a good review of methods to solve hub-location problems.

In this study, we deploy sensors and communication hubs along the barrier region using hub-and-spoke topology. The presence of a hub-and-spoke topology brings up the question of how to allocate the budget between hubs and sensors, and how to deploy both hubs and connected sensors to maximize the coverage under certain topological constraints.

Figure 1 shows an exemplary belt-shaped barrier. The squares represent candidate sensor and/or hub locations. Allocated sensors are shown as circles and allocated hubs as triangles. There are two types of sensor, represented by dotted-blue and solid-red circles. In the real-world, the examples of sensor type include radars, optic sensors, thermal imaging cameras and similar intrusion detection systems that are used to observe the physical space. There are two types of target. In the real-world, the examples of target type include humans (terrorists, smugglers), vehicles and animals. Each dashed arrow pointing down represents discretized possible target paths. We show two examples of a sensor’s minimum and maximum communication ranges, shown as concentric circles where the inner and outer circles represent the minimum and maximum ranges, respectively.

In this study, we consider allocating a number of sensors of different types to a set of pre-determined candidate locations within a barrier to maximize the level of coverage (which we will define later in the objective function) along the barrier region against targets.



Figure 1: An exemplary WSN belt-shaped line-barrier-problem setup.

## INLP formulation

We now present the details of our INLP formulation. The model will use the following sets, indices, and decision variables.

### Sets, Parameters and Decision Variables

We use the following notation to represent the sets and indices:

|  |  |  |
| --- | --- | --- |
|  | : | set of candidate locations for sensors and hubs |
|  | : | set of target paths |
|  | : | set of sensor types |
|  | : | set of target types |
|  | : | subset of candidate locations that are appropriate for sensor type *s* |
|  | : | subset of candidate locations that are appropriate for hubs |

We denote the set of sensor types as *S*, indexed by , the set of target types as *T*, indexed by , and the set of candidate locations as *I*, indexed by . Note that, in our model we assume that only a subset of locations in set *I* are appropriate for sensors of type *s* and hubs, and we denote the subsets of these locations as  and , respectively. We discretize the barrier segment into small intervals that separate target paths and denote the set of target paths as *P*, indexed by . In real life, intruders are highly likely to follow the shortest path across a barrier line or a belt region (Chen et al. 2007). Thus, it would be a realistic assumption to model intruder paths as straight lines perpendicular to the barrier line.

The decision variables in our study are:

|  |  |  |
| --- | --- | --- |
|  | = |  |
|  | = |  |
|  | = |  |
|  | = | number of sensors of type *s* (integer) |
|  | = | number of hubs (integer) |

The list of parameters used in our mathematical model is given as follows:

|  |  |  |
| --- | --- | --- |
| *B* | = | total budget ($) |
|  | = | cost of a sensor of type *s* ($) |
|  | = | cost of a hub ($) |
|  | = | weight of a target type *t* |
|  | = | probability that a target is of type *t* |
|  | = | reliability of a sensor of type *s* |
|  | = | hub capacity (maximum number of sensors that a hub can be assigned to) | |
|  | = | lateral distance between location *i* and target path *p* |
|  | = | distance between locations *i* and *i'*. |
| *L, W* | = | length and width of the barrier region |
| , | = | maximum and minimum communication ranges of a sensor |
|  | = | detection-performance coefficient for sensor-type *s* and target-type *t* |
|  | = | probability of detecting type *t* target following path *p* by a sensor of type *s* located at *i* |
|  | = | penalty parameters (big numbers) |

*B* is the total budget that can be allocated to hubs and sensors (i.e. ). We assign each target type *t* with a weight *vt* that reflects its threat level or relative value. Additionally, we denote the probability that a target is of type *t* by the parameter. We associate each sensor type *s* with a reliability coefficient  such that . Reliability coefficient values 0 and 1 reflect totally unreliable and totally reliable sensors, respectively, whereas values in-between denote the probability that a sensor does not fail during a mission. This extension makes it essential to design networks which also consider the different failure probabilities of each sensor type, so as to achieve more efficient and reliable barriers. As a simplifying assumption, we suppose that hubs are more expensive than sensors and are completely reliable. We also assume that each hub has a capacity  which represents the maximum number of sensors that can be assigned to that particular hub. A lateral distance  represents the shortest distance from sensor at *i* to the path *p* traversed by a target, i.e., the “distance of closest approach” (Arnold and Bram, 1962).

The deployment of sensors and hubs in hub-and-spoke topology is affected by the communication range of sensors. In this paper, we assume that all sensors have a fixed maximum and minimum communication range denoted by  and , respectively. Under this simplifying assumption, a sensor can communicate with a hub if it lies within the pre-determined maximum communication range *r*+. *r*+ represents the limit on the transmission range in the open space. This range is affected by the sensor’s output power and hub’s receiving antenna gain (which affects the received signal quality). Therefore, in our model, communication between a sensor and hub is not possible if the distance between them is larger than *r*+. We further account for interference phenomena and assume that neighbouring sensors suffer from interference and cannot communicate with their designated hubs if the distance between them is less than the pre-determined minimum communication range . In other words, if two sensors are adjacent and transmit simultaneously, their messages interfere and cannot be processed correctly (resulting in bit errors). Such a condition among sensors in a WSN has a degrading effect on system reliability. Hence, our model will ensure that the distance between a sensor and its designated hub will always be less than  (healthy communication requirement) and no other sensor is located within a radius  around a sensor (to prevent interference). We add the following sets in our model:

|  |  |  |
| --- | --- | --- |
|  | : | subset of locations which lie within the maximum communication range of a sensor located at *i*, i.e. |
|  | : | subset of locations which lie within the minimum communication range of a sensor located at *i*, i.e. |

In most coverage problems, the detection level of a target is simply computed by considering the distance between the target and its closest sensor, i.e. only the closest sensor that can detect it. In reality, a region (or a target) can be monitored (covered) cooperatively by multiple sensors, such that each sensor contributes to the detection performance of a target (Karatas, 2017). In this study, assuming independent detections, we model the overall detection probability of a target attempting to penetrate the barrier by allowing the possibility of cooperative coverage. Let  denote the subset of candidate locations installed with a sensor, and let  be the probability of detecting a target with a sensor at *i*. Then, the overall non-detection (survival) probability of a target is the product of individual non-detection probabilities  for all sensors. Finally, the cumulative detection probability, *PD*, for that target is given in equation :



There are basically two different types of sensing models adopted by researchers in the literature: disc-sensing model, and probabilistic sensing model (He et al. 2013). The former, also known as the binary sensing model or definite-range model, assumes that the detection region of a sensor is a disk region with a radius of the deterministic sensing range. Detection is only possible within this region. In real life, the detection probability of a sensor generally depends on the sensor-target distance. Hence, the latter model is more realistic due to using a probability function to determine coverage or detection. There are different probabilistic sensing model types in the literature, including exponential, polynomial, cubic and Fermi-type.

In this study, we consider a probabilistic sensing model in which the probability of detection is a non-increasing function of distance. We employ the widely used exponential function given in equation . We further assume that each target type *t* has a unique detection probability function with respect to a sensor   
type *s*. We model these differences with respect to detection performance by assigning a unique *αs,t* value to each (*s*,*t*) pair. Thus, we calculate, the detection probability of a target type *t* following path *p* with a sensor of type *s* located at *i* as:



where  is the lateral distance between location *i* and target path *p*.

### Objective Function

The objective function seeks to maximize the total weighted-detection probability of targets attempting to penetrate the barrier. Note that parameters , , and  are used in the objective function to account for the sensor reliability, target weight and probability extensions, respectively. The cooperative coverage and probabilistic sensing concepts are also implemented in the function as described in equations and .



### Constraints

































Constraint ensures that the total cost of sensors and hubs does not exceed the allocated budget. Constraint sets and limit the number of located sensors and hubs to  and respectively. Constraint set ensures that at most one sensor or hub can be installed at a location. Constraint set ensures that a sensor can only be assigned to an activated hub, and the total number of sensors assigned to a particular hub cannot exceed the hub’s capacity. Constraint set , on the other hand, dictates that each sensor is assigned to a hub if it lies within the sensor’s maximum communication range and cannot be assigned to more than one hub. Additionally, in accordance with our assumptions in section 3, constraint set guarantees that each sensor is assigned to its closest hub. Constraint set ensures that no other sensor is located within the minimum communication range of a sensor at *i*. In their study, Camm et al. (1990) prove that setting the penalty parameter – such as *M*1 and *M*2 used in constraint sets and – to a tight value helps in reducing the CPU time a solver needs to find a solution. In constraint set , to maintain feasibility, when , the left-hand side of the inequality should always be less than the value of . Hence, we set  to the maximum possible distance between two locations in the region, i.e.  For constraint set , when the left-hand side equals , it should always be greater than the maximum number of sensors that can be located in a circular region of radius . Hence, although unlikely, we set  to the total number of sensors that can be purchased by budget *B*, i.e. . Constraint set ensures that at least one sensor should be assigned to an active hub. Business rules dictate that specific sensor types and hubs are not allowed at certain locations. Hence, constraint sets and ensure that a sensor or hub is not assigned to an unsuitable candidate location. Constraint sets – declare the variable domains.

## ILP formulation

In this section, we approximate our INLP model by an ILP version to attain a globally optimal solution to the aforementioned barrier problem. To this end, we adopt a special linearization technique as used by Morton et al. (2007), Salmerón (2012) and Karatas (2017, 2018). This technique maps the defined location problem to a network-like structure by representing each candidate location as a node and the survival probability gained by locating/not locating a sensor to that node as arcs between consecutive nodes. This is achieved by introducing new parameters, decision variables, and constraints, as explained below.

Consider a network composed of |*I*|+1 nodes ordered consecutively, and |*S*| positive and one negative labelled arcs between each neighbour node pair (*i*, *i*+1). Let each node in the network denote an arbitrary candidate sensor location . The arcs symbolize a measure of flow associated with the overall survival probability of a target *t* taking a path *p*. Figure 2 illustrates this network which represents a flow for a single path *p*. In our ILP formulation, we ensure that a positively labelled arc *s* between consecutive nodes (*i*, *i*+1) gets a positive value only if a sensor of type *s* is installed at location *i*. On the other hand, a negatively labelled arc gets a positive value if no sensor is placed at *i*. The total amount of flow leaving node *i* symbolizes the total survival probability of a target following path *p* from the first *i* locations. Hence, starting from node 1 and proceeding towards node |*I*|+1, this flow either decreases (as sensors are placed at certain locations) or remains unchanged. It is ensured that, in an optimal solution, flow can be positive in exactly one of the arcs leaving from node *i* to node *i*+1. This is achieved by the decision variables  and  which are called as the “*positive* *transitory survival probabilities*” and “*negative* *transitory survival probabilities*”, respectively. In other words, these variables are used to represent the *amount of flow* across the arcs.

A change in the total flow is accomplished by scaling it down by a factor of . If , then the positive flow leaving node *i* is scaled down by a factor of . We call these “*scale factors”* for the transitory survival probabilities. To be specific,  is the probability that a type *s* sensor located at *i* does not detect target type *t* following path *p*, i.e.  If the candidate location corresponding to a node *i* does not have a sensor installed at it, the total amount of flow remains the same, i.e.  In the ILP formulation we compute the overall survival probability of a target of type *t* following path *p* by the variable . Note that node |*I*|+1 is imaginary, it represents the overall survival probability of a target of type *t* following path *p*.



Figure 2. Representation of the scaled transitory survival probability flow between candidate locations for a single target type *t* following path *p*. Positive and negative arcs are denoted by solid and dashed lines, respectively. Each disk represents a candidate location.

### Additional parameters and decision variables

Implementing the method described above, we have added the following decision variables:

|  |  |  |
| --- | --- | --- |
|  | = | positive transitory survival probability of a target type *t* following path *p* between the *i*th and (*i*+1)th locations for sensor type *s*. |
|  | = | negative transitory survival probability of a target type *t* following path *p* between the *i*th and (*i*+1)th locations. |
|  | = | overall survival probability of a target of type *t* following path *p*. |

We also define the following parameters that will be used to scale the total amount of flow in the network-like structure of the problem:

|  |  |  |
| --- | --- | --- |
|  | = | scale factor for the transitory survival probability of target type *t* following path *p* if a sensor of type *s* is placed at the *i*th location. |
|  | = | scale factor for the transitory survival probability of target type *t* following path *p* if no sensor is placed at the *i*th location. |

### Objective Function

The objective function is the same as , i.e. maximizes the total weighted-detection probability of targets, except that we now represent the cumulative detection probability of a type *t* target following path *p* as one minus the overall survival probability, i.e. .



### Additional Constraints

We have added the following constraints to constraints -:















We name the constraint sets , , and as the flow initialization, flow scale, survival probability and technical constraints, respectively. To be specific, constraint set initializes the survival probability flow for each target *t* and path *p* by setting their respective sums of negative and positive transitory survival probabilities to one. Note that, in this constraint, the transitory survival probabilities are summed only for the arcs between nodes 1 and 2 for each path *p* and target type *t*. This also ensures, as the flow pertaining to the survival probability is scaled down through the network, the sum of final detection and survival probabilities of a type *t* target following path *p* is also exactly one. As a specific case, this constraint also ensures that it is not possible to detect a target if no sensor is installed in the region. Next, constraint set sets the value of the entering scaled flow equal to the unscaled flow leaving node *i* for each target *t* and path *p*. The left-hand side of this constraint represents the sum of the scaled transitory survival probability of a type *t* target following path *p* between nodes *i* and *i*+1. Next, on the right-hand side of the equation, this value is set to the sum of the unscaled transitory survival probability (of the same target type andpath) between nodes *i*+1 and *i*+2. So, for a specific node *i*, this constraint assures that the total entering unscaled survival probability is equal to the leaving scaled probability. For each target type *t* following path *p*, constraint set calculates the overall survival probability  by summing the total scaled flow leaving node |*I*| and entering imaginary node |*I*|+1. Constraint set is a technical constraint and it ensures that the positive transitory survival probability value  is zero if no sensor is placed at location *i*, i.e. Conversely, it also assures that  takes the value 1 if . Finally, constraint sets - declare the variable domains for the additional decision variable.

## Optimisation-via-Simulation formulation

This barrier coverage problem can be represented as an optimisation-via-simulation (OvS) problem. The difference is that in OvS, the objective function is not calculated using equation , instead it is estimated from a simulation model. Strictly speaking, OvS is not necessary for this problem because we can compute the objective function value simply using equation . However, as we will demonstrate it later, when we relax the assumptions of this problem such that it becomes analytically intractable, a simulation model can provide an estimation of the value. Hence, OvS provides an alternative method for analytically intractable problems. Furthermore, by starting with a simplified OvS model (or base model) that can be validated analytically, we have more confidence in the correctness of the base model before we add more realism to, or relaxing some of the assumptions of the base model.

The simulation model used in OvS is an Agent-Based Simulation (ABS) model. The ABS model is formed by two agent types: sensor and target. All sensors are instantiated at the start of a simulation run and remain active until the last target is detected or has crossed the barrier undetected. There is only one target at any time during the simulation. When a target is detected or has crossed the barrier, a new target is created. The cycle continues until a fixed number of targets has been simulated.

There are  sensors of type *s* and the behaviour of each sensor is shown in Figure 3. For each target, each sensor will check if it is operational based on its reliability *βs* (line 2). If the sensor is operational, when there is a target that has not been detected and has not crossed the barrier (line 3), the sensor will try to detect the target with a probability that is calculated using equation (lines 4 and 5). Hence, all sensors work together to detect a target. If this sensor detects a target, then the target will be flagged as detected (line 6). This flag will also be seen by other sensors. It should be noted that for the experiments in Sections 4.1 and 4.2, we use the same assumptions as in the ILP and INLP models, i.e. the distance function returns the lateral distance which is the distance between the *x* coordinates (i.e. we ignore the *y* coordinates). For the experiments in Section 4.3, we relax the assumption by using the Cartesian coordinate system and distance.

1. **foreach** target t **do**

2. **if** (U(0,1) < β) **then** // the sensor is operational

3. **while** (t.detected == **false**) **and** (t has not crossed the barrier) **do**

4. d = distance(this\_sensor\_location, t.location)

5. pd = exp(-α(this\_sensor\_type, t.type)\*d) // see equation 2

6. **if** (U(0,1) < pd) **then** t.detected = **true**

7. **end**

8. **end**

9. **end**

Figure 3: The behaviour of agent sensor in the ABS model

Figure 4 shows how the targets are simulated. To estimate the objective function (equation ) we need to collect samples from a number of targets (lines 1, 2 and 21). Line 3 generates a target and line 4 sets its initial status as undetected. Lines 5 to 9 set the target type based on the proportion *λ*. Line 10 sets the initial location of the target. As mentioned earlier, for the experiments in Sections 4.1 and 4.2, we consider only the lateral distance between sensors and a target. Hence, line 10 simply places the target at a random location on the *x* axis. For the experiments in Section 4.3, we place the target along the starting line   
*y* = 40 (see Figure 6). The target will try to reach the nearest point along the destination line *y* = 0.

While the target has not been detected and has not reached the destination line (line 11), the target will keep moving (line 16). Before a target moves, all sensors will attempt to detect it (see Figure 3). If at least one of the sensors detects it (line 13), we will increase the number of detected targets (line 14). For the experiments in Sections 4.1 and 4.2, the move() function simply moves that the target pass the barrier (i.e. all sensors have only one chance to detect the target). For the experiments in Section 4.3, the target will move along a straight-line that forms the shortest distance between the initial location and the destination line. Lines 11 to 18 is repeated until the target is detected or successfully crosses the barrier undetected. In the latter case, the number of undetected target is increased (line 19).

1. i\_target = 0

2. **while** (i\_target < n\_targets) **do**

3. generate target t

4. t.detected = **false**

5. **if** (U(0,1) < λ) **then** // two target types

6. t.type = 0

7. **else**

8. t.type = 1

9. **end**

10. t.location = initial\_location()

11. **while** (target t has not crossed the barrier) **and** (t.detected == **false**) **do**

13. **if** t.detected == true **then**

14. increase the number of detected targets

15. **else**

16. t.location = move()

17. **end**

18. **end**

19. **if (**t.detected == **false**) **then** increase the number of undetected targets

20. remove target t

21. i\_target = i\_target + 1

22. **end**

Figure 4: The behaviour of agent target in the ABS model

The OvS algorithm we use in this paper is shown in Figure 5. Line 1 computes the minimum number of hubs  (i.e. when each hub is connected to the maximum number of sensors) and the maximum number of hubs  (i.e. when each hub is connected to a sensor). This line also guarantees that the budget constraint is met.

The main idea of Line 2 is to identify areas (or partitions) in the solution space that are likely to contain a good solution. Each partition is defined by the number of hubs () and the numbers of the two sensor types (and ) that are feasible under the hub’s capacity constraint  Hence, a partition  contains all solutions that are formed by  hubs,  sensor type-1 and  sensor type-2. To decide the promising partitions, we use the following heuristics. For a given number of hubs (), all feasible combinations of sensor type-1 () and sensor type-2 () with the highest total number of sensors  will be considered as promising partitions. The reason is that for a given number of hubs, the more sensors we have (subject to hub capacity), the more likely we can cover more areas. We call any other feasible combinations with the same  that has fewer total number of sensors as “dominated” and they will not be used as promising partitions. All promising partitions form the promising solution space ℘ (line 2.b.ii).

Before proceeding with the OvS algorithm and our simulation runs, we find it useful to define an additional set and a number of parameters:

|  |  |  |
| --- | --- | --- |
|  | = | Set of feasible solutions |
|  | = | Minimum number of hubs |
|  | = | Maximum number of hubs |
|  | = | Maximum number of sensor type *s* |
|  | = | The best objective function value so far |
|  | = | The mean objective function value of solution *j* |
|  | = | A set of promising partitions |
|  | = | A partition that contains solutions with  number of hubs,  number of sensor type-*s* and  number of sensor type-*s'* |
|  | = | The mean of all solutions in partition ; it is an indicator of how promising the partition is |
|  | = | The memory to record the best solution and its estimated objective function value in partition |
|  | = | The number of global runs so far |
|  | = | The maximum number of global runs |
|  | = | The objective function value estimated by ABS model |
|  | = | The number of detected targets in path *p* |

|  |
| --- |
| 1. Let  and  and  2. For  a. Let   1. For  to 0 do   i. Let   1. For  to 0 do   *(1)* If  is feasible and not dominated then create a partition  and  3. Set , ,  for all  4. While ( and no improvement in  for all ) do  a. For in  do  i. Randomly select |*J*| feasible solutions in , where |*J*| =   1. Let  for  , if *j* has been visited then refine 2. Record  in 3. if  then  and 4. Let . be the mean of  for all  in 5. Set |

Figure 5: OvS Algorithm

Line 3 initialises the variables used to record the best solution and the stopping condition which will be used in the main OvS loop (i.e. line 4). In each iteration, feasible solutions are sampled from each partition (4.a.i). At this stage, feasibility is determined by the interference and maximum communication distance ( and  respectively). When a feasible solution cannot be found, it will proceed to the next partition. The ABS model is run several times in line 4.a.ii for each feasible solution found in step 4.a.i to estimate *z*. If a solution has been visited (i.e. run) before, its estimated value is refined. The visited solutions and their estimated values are stored in the memory (line 4.a.iii). Line 4.a.iv records the new best solution (if any). Line 4.a.v calculates the mean of the estimated value for each partition. This value will be used to determine the number of feasible solutions that will be sampled in each partition in the next iteration. The algorithm will spend more time sampling solutions from the most promising partition. The iteration ends when the computing budget has been exhausted or the improvement to the mean of the estimated solution in each partition is too small. If the computing budget is too high, the execution will take too much time. If it is too low, the search for better solution is terminated too early leading to a lower quality solution. The best solution is stored in *j\** and its expected objective function value is . One main benefit of OvS is that the simulation model used in the estimation (line 4.a.ii) can easily be changed without affecting the overall OvS algorithm, as demonstrated in Section 5.3 where we relax some of the model assumptions.

# EXPERIMENTS and discussion

This section starts with the validation of our models (section 4.1). The ILP and INLP models are implemented in the General Algebraic Modeling System (GAMS©) 24.2.3 environment and solved by CPLEX 12.2.0.2 and DICOPT, respectively. The OvS model is implemented using Python 3.6.1 and NumPy 1.13.3. Based on a few trials,  gives us a good compromise. The value of  depends on the variability of the outputs. Techniques such as indifference zone (Kim and Nelson 2006) may be applicable to find the best , but further research is needed. We use the three models (ILP, INLP and OvS) to cross-validate each other and ensure that their results are consistent and the simulation model used in OvS is valid. This is followed by two experiments. The first experiment (section 4.2) is to support our primary contribution in building the INLP model by evaluating its performance against ILP in terms of the quality of their outputs and computation times. In addition, we also compare the performance of the OvS model to demonstrate that the INLP is better than the OvS model. The second experiment (section 4.3) is to support our secondary contribution, i.e. to demonstrate that, in certain cases, OvS is useful, for example, when the assumptions are relaxed so that the targets can move along a continuous path at different speeds depending on the path; and they can also learn from other targets.

## Validation of the three models

Table 1: Fixed parameter values

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| |*P*| | 100 |
| |*S*| | 2 |
| |*T*| | 2 |
| *B* | 50 |
|  | 20 |
|  | 5 |
|  | 3 |
|  | 20 |
|  | (10,15) |
|  | (0.5,0.8) |
|  | (0.3,0.7) |
|  | (0.15, 0.85, 0.35, 0.65) |
| (*L*,*W*) | (100, 30) |

For validation, we use nine test cases that are designed to reflect how different budget levels and sensor reliabilities affect the objective function. We vary the budget levels between low (*B*=50), medium (*B*=100) and high (*B*=150) and levels of sensor reliability are high (*β* = {0.9, 0.95}), medium (*β* = {0.7, 0.85}) and low (*β* = {0.5, 0.75}), while keeping the remaining parameters fixed. The values of fixed parameters are indicated in Table 1. The candidate locations for sensors and hubs are given in Table 2 and plotted in the barrier region in Figure 6. First, we solve the test cases using ILP and INLP to find optimum solutions. Table 3 shows that ILP produces better solutions because ILP guarantees a global optimum solution. Next, we simulate the solutions produced by the ILP and INLP models using the ABS model in our OvS to cross-validate our models. We use ten replications for each test case. Table 3 shows that our ABS model can estimate *z* well.



Figure 6: Candidate locations for sensors and hubs in the barrier region

Table 2: Candidate locations (*xi*, *yi*) for sensors and hubs.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***i*** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| ***xi*** | 39.52 | 65.91 | 74.86 | 23.49 | 99.09 | 76.03 | 71.56 | 9.77 | 55.29 | 51.39 |
| ***yi*** | 6.20 | 21.61 | 14.02 | 15.71 | 11.63 | 29.80 | 16.21 | 6.65 | 19.63 | 20.89 |

Table 3: Validation results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | | **Model INLP** | | **Model ILP** | |
| **Budget** | **Sensor reliability** |  | (95% CI) |  | (95% CI) |
| Low | Low | 4.62 | 4.64 ± 0.08 | 4.76 | 4.74 ± 0.05 |
| Low | Medium | 6.34 | 6.32 ± 0.07 | 6.54 | 6.54 ± 0.08 |
| Low | High | 7.92 | 7.86 ± 0.09 | 8.25 | 8.23 ± 0.09 |
| Medium | Low | 8.27 | 8.25 ± 0.12 | 8.52 | 8.47 ± 0.08 |
| Medium | Medium | 10.79 | 10.64 ± 0.10 | 10.85 | 10.83 ± 0.11 |
| Medium | High | 13.32 | 13.36 ± 0.08 | 13.33 | 13.39 ± 0.13 |
| High | Low | 10.91 | 10.84 ± 0.12 | 11.22 | 11.09 ± 0.13 |
| High | Medium | 12.49 | 12.43 ± 0.06 | 12.71 | 12.54 ± 0.13 |
| High | High | 15.06 | 14.93 ± 0.12 | 15.19 | 15.14 ± 0.06 |

## Performance evaluation of the three models

After validating our models, we can confidently compare their performance. In this experiment, we vary the number of possible locations |*I*| from 10 to 50 and the budget levels between high (*B*=50), medium (*B*=100) and low (*B*=150) and keep the remaining parameters fixed (fixed parameters are shown in Table 1). For each test case (i.e. a row in Table 4), we create ten sets of locations to evaluate if there is a location effect. We present the average performance of the ten location sets in Table 4. The experiments are run on a PC (Intel Core i5-3330S CPU 2.70 GHz processor with 6.00 GB memory and Microsoft Windows 10 64-bit operating system).

The result shows that computing time is a function of the number of locations (|*I*|) and budget (*B*). An increase in the number of locations and budget level increases the number of possible solutions, which makes the problem size bigger. The ILP model produces globally optimum solutions but it requires long computing times, especially as the problem size gets bigger. In fact, for bigger problem sizes (from |*I*|=30 and *B*=150 onwards), our ILP model cannot solve the problem in less than one hour so we limit the computing time to one hour. On the other hand, our INLP model is very fast and can produce solutions that are very close to the ILP solutions (no more than 5% difference, see column 10). This means that our INLP model is more practical for solving this barrier-coverage problem.

Our OvS model can produce solutions that are relatively close to the ILP solutions (within 10% difference, see column 11). In fact, for smaller problem sizes, the OvS may even outperform the INLP (see column 9). However, our INLP model is significantly faster than our OvS model and for bigger problem sizes the INLP results dominate the OvS results. If we compare the ILP and OvS models, the ILP model is significantly faster than OvS for smaller problem sizes. However, as the problem size increases, the computing time for ILP increases faster than the computing time for OvS. As we can see from the results, for a problem size of |*I*|=30 or bigger, ILP is slower than OvS.

This experiment shows our proposed INLP model performs very well (i.e. the quality of the solution is close to the global optimum solution produced by ILP and significantly faster). The experiment also shows that OvS is not practical in this setting.

Table 4: Performance comparison results

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Average Computing Time (seconds)** | | | **Average Objective Function Values** | | | | | |
| |*I*| | Budget | ILP | INLP | OvS | ILP | INLP | OvS | Frequency OvS better than INLP\* | INLP / ILP (%) | OvS / ILP (%) |
| 10 | 50 | 1.61 | 0.42 | 32.94 | 6.00 | 5.94 | 5.77 | 6/10 | 99% | 96% |
|  | 100 | 4.57 | 0.36 | 281.08 | 10.03 | 9.86 | 9.85 | 5/10 | 98% | 98% |
|  | 150 | 3.22 | 0.26 | 242.85 | 11.49 | 11.27 | 10.95 | 0/10 | 98% | 95% |
| 20 | 50 | 6.03 | 0.30 | 38.02 | 6.61 | 6.44 | 6.42 | 4/10 | 97% | 97% |
|  | 100 | 86.02 | 0.33 | 385.04 | 12.42 | 12.08 | 11.54 | 1/10 | 97% | 93% |
|  | 150 | 321.54 | 0.55 | 512.85 | 17.17 | 16.72 | 16.16 | 1/10 | 97% | 94% |
| 30 | 50 | 17.23 | 0.63 | 29.62 | 6.67 | 6.34 | 6.43 | 5/10 | 95% | 96% |
|  | 100 | 515.29 | 0.64 | 475.85 | 12.74 | 12.58 | 11.63 | 0/10 | 99% | 91% |
|  | 150 | 3600.21 | 2.33 | 524.31 | 17.70 | 17.44 | 16.07 | 0/10 | 99% | 91% |
| 40 | 50 | 49.18 | 1.03 | 44.99 | 6.73 | 6. 57 | 6.45 | 1/10 | 98% | 96% |
|  | 100 | 2,538.07 | 2.12 | 486.14 | 12.82 | 12.65 | 11.87 | 1/10 | 99% | 93% |
|  | 150 | 3,600.24 | 1.65 | 565.00 | 17.79 | 17.62 | 16.13 | 0/10 | 99% | 91% |
| 50 | 50 | 106.16 | 1.62 | 40.07 | 6.76 | 6.43 | 6.41 | 5/10 | 95% | 95% |
|  | 100 | 3,600.19 | 3.14 | 543.49 | 12.88 | 12.70 | 11.93 | 0/10 | 99% | 93% |
|  | 150 | 3,600.16 | 2.52 | 600.30 | 17.80 | 17.71 | 16.16 | 0/10 | 100% | 91% |

(\*) The OvS solutions are dominated by ILP solutions (0/10)

## Intelligent moving targets

Under the assumptions we use in Section 4.2, the experiment has shown that OvS is not practical because the model is solvable using INLP, which is faster than OvS and gives better results in the majority of cases, especially for bigger problem sizes. The main benefit of OvS is that once we have the model, it is relative easy to relax the assumptions in the OvS model in comparison to INLP. In fact, in complex cases, developing an INLP model is extremely challenging.

For example, suppose we want to relax the lateral distance assumption (i.e. sensors have only one chance to detect targets as if the targets can pass the barrier within one time unit) so to consider a more realistic scenario in which targets need time to move along the straight line in the Cartesian coordinate. Hence, sensors will have more chances to detect the targets. We also relax the assumption that the paths are homogeneous (e.g. some paths are more difficult). In the experiment, we consider any path with *x* coordinate ≥ 50 to be easier so that the targets can move twice as fast. We call this the “moving target” scenario. To implement this, we only need to do two changes to the behaviour in Figure 4. First, the function initial\_location() in line 10 places a target at a random integer *x* coordinate. The second change is the function move() in line 16 reduces the *y* coordinate of a target by one if *x* < 50 (i.e. the target moves downward in a straight line, see Figure 5); otherwise, the *y* coordinate is reduced by two (i.e. the targets move twice as fast).

In the second scenario, we add one more assumption to moving target scenario that the targets also share intelligence and choose their paths based on information they have gathered (e.g. detected targets reported by the authorities or media). We call this scenario “moving intelligent target”. To implement this, we define and maintain a vector *D* of size |*P*| where *Dp* denotes the number of detected targets in path *p* where *p*∈*P*. In this experiment, all targets have access to *D*. When we first run the simulation, we set  As more and more targets are simulated, *D* vector will show which *x* coordinates (paths) have the higher number of detections. Consequently, targets that are simulated later have a better knowledge than those simulated earlier by choosing the smallest *Dp* (path p with the smallest number of detection).

To show the impact of the new assumptions, we use the optimum solutions produced by ILP from Table 4 for cases where |*I*|=20 and *B*={50, 100, 150}. The reason is that, in this setting, ILP can run until completion so the ILP results are globally optimum. We run the ABS model to estimate the objective function values of the ILP solutions under three settings: (1) strong assumptions as used in ILP (we refer to this as the “base case” in Figure 7), (2) moving targets and (3) moving intelligent targets.

The results show that the difference between the base case and the other two cases (moving targets and moving intelligent targets) is significant. When we relax the assumptions on target movements, the expected detection probability is significantly higher. This is because the target speeds are relatively slower than the time between sensor pings. Hence, the sensors have more chances to detect targets. When we add intelligence into the targets, the performance drops significantly because the targets are getting smarter as more and more targets attempt to break the barrier.





Figure 7: The impact of more realistic assumptions

Given the fact that the difference in the objective function values are significant, it shows that the ILP solutions are not necessarily optimal under the more relaxed assumptions. Hence, model reformulation is needed. It is challenging to reformulate the ILP or INLP models to address moving targets and moving intelligent targets where the speeds of the targets vary depending on the paths the targets take. However, it is relatively easy to modify the OvS model. This is where OvS model can complement the use of INLP and ILP in solving barrier coverage problem.

# CONCLUSION

We have proposed three models to address the optimum deployment of wireless sensor networks (WSN) that form a barrier against illegal intruders (or targets). The three models are Integer Non-Linear Programming (INLP), Integer Linear Programming (ILP) and Optimisation-via-Simulation (OvS). The following factors have been included in the models: sensor types, target types, probabilistic detection function, sensor reliability, communication range, communication interference, network topology and budget constraints, which make our model the most elaborate that has been reported in the literature.

We have also demonstrated that although the ILP model can produce global optimum solutions, due to its computing time, it is only practical for small problems. On the other hand, our INLP model is fast and produces solutions that are close to the ILP’s global optimum solutions. We have further shown that when the assumptions are relaxed to address intelligent targets that can move freely across the barrier and the movement speed depends on the path taken, the ILP and INLP models need to be reformulated because they are no longer optimum due to bias in calculating objective function values. In this setting, we have shown that OvS has more benefit because it is relatively easy to relax the assumptions in the model. Hence, OvS has the potential to contribute to research into the WSN coverage problem.

Hence, two contributions are made. First, our ILP, INLP and OvS models are more elaborate than other models in the literature. The second contribution is that this is the first work that demonstrates how OvS can be used to solve a barrier-coverage problem and its benefit in handling more realistic assumptions. We are currently working on improving the performance of our OvS algorithm to handle bigger problem sizes using Nested Partition (Pitchitlamken and Nelson 2003). We also plan to extend our model to include communication protocol layers so that we can evaluate the performance of various communication protocols (e.g. contention-based, contention-free, hybrid MAC protocols) running on top of the network structure produced using the model in this paper.

# REFERENCES

Abo-Hamad W and Arisha A (2011). Simulation–Optimisation Methods in Supply Chain Applications: A Review. *Irish Journal of Management*, 30 (2): 95 – 124.

Ahmed M A and Alkhamis T M (2009). Simulation optimization for an emergency department healthcare unit in Kuwait. *European Journal of Operational Research*, (198): 936 – 942.

Alrabghi A and Tiwari A (2015). State of the art in simulation-based optimisation for maintenance systems. *Computers & Industrial Engineering*, 82: 167 – 182.

Amaran S, Sahinidis N V, Sharda B, and Bury S J (2016). Simulation optimization: A review of algorithms and applications. *Annals of Operations Research*, 240(1): 351-380.

Arnold, R. D., & Bram, J. (1962). Estimating the Lateral Range Curve from Observed Detection Ranges, Interim Research Memorandum IRM-27, Operations Evaluation Group, Center for Naval Analyses, Washington, D.C.

Arora A, Dutta P, Bapat S, Kulathumani V, Zhang H, Naik V, Mittal V, Cao H, Demirbas M, Gouda M, Choi Y, Herman T, Kulkarni S, Arumugam U, Nesterenko M, Vora A and Miyashita M (2004). A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5): 605-634.

Avramidis AN, Chan W, Gendreau M, L’Ecuyer P and Pisacane O (2010). Optimizing daily agent scheduling in a multiskill call center. *European Journal of Operational Research*, 200: 822 – 832.

Basagni S, Conti M, Giordano S and Stojmenovic I (2004). *Mobile ad hoc networking*. John Wiley & Sons.

Benkoczi R, Friggstad Z, Gaur D and Thom M (2015). Minimizing total sensor movement for barrier coverage by non-uniform sensors on a line. In: *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pp. 98-111. Springer, Cham.

Camm JD, Raturi AS and Tsubakitani S (1990). Cutting big M down to size. *Interfaces*, 20(5):61-66.

Cardei M and Wu J (2006). Energy-efficient coverage problems in wireless ad-hoc sensor networks. *Computer communications*, 29(4):413–420.

Chang KH and Lin G (2015). Optimal design of hybrid renewable energy systems using simulation optimization. *Simulation Modelling Practice and Theory*, 52: 40–51.

Chen A, Kumar S and Lai TH (2007). Designing localized algorithms for barrier coverage. In: *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pp. 63-74. ACM.

Chen A, Lai TH and Xuan D (2008). Measuring and guaranteeing quality of barrier-coverage in wireless sensor networks. In: *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pp. 421-430. ACM.

Chen J, Wang B and Liu W (2015). Constructing perimeter barrier coverage with bistatic radar sensors. *Journal of Network and Computer Applications*, 57:129-141.

Cheng CF, Wu TY and Liao HC (2014). A density-barrier construction algorithm with minimum total movement in mobile WSNs. *Computer Networks*, 62:208-220.

Contreras I (2015). Hub location problems. In: *Location science,* pp. 311-344. Springer International Publishing.

De Keizer M, Haijema R, Bloemhof J M, and Van Der Vorst J G (2015). Hybrid optimization and simulation to design a logistics network for distributing perishable products. *Computers & Industrial Engineering*, 88: 26-38.

Fan G and Jin S (2010). Coverage problem in wireless sensor network: A survey. *Journal of Networks*, 5(9):1033-1040.

Farahani RZ, Hekmatfar M, Arabani AB and Nikbakhsh E (2013). Hub location problems: A review of models, classification, solution techniques, and applications. *Computers & Industrial Engineering*, 64(4):1096-1109.

Fu MC (2002). Optimization for simulation: theory vs. practice. *Journal on Computing*, 14:192–215.

Fu MC (2015). *Handbook of Simulation Optimization*. New York: Springer-Verlag.

Gage DW (1992). Command control for many-robot systems. In: *Proceedings of AUVS-92,* Vol. 10(4), pp. 28–34*.* Naval Command Control and Ocean Surveillance Center Rdt and E Div, San Diego CA.

Galyaev AA and Maslov EP (2011). On the border patrolling problem. *Journal of Computer and Systems Sciences International*, 50(5): 837.

Gong X, Zhang J, Cochran D and Xing K (2016). Optimal placement for barrier coverage in bistatic radar sensor networks. *IEEE/ACM Transactions on Networking*, 24(1):259-271.

He S, Chen J, Li X, Shen XS and Sun Y (2012). Cost-effective barrier coverage by mobile sensor networks. In: *Proceedings of INFOCOM*, pp. 819–827.

He S, Chen J, Li X, Shen XS and Sun Y (2014a). Mobility and target prior information improving the barrier coverage of sparse sensor networks. *IEEE Transactions on Mobile Computing*, 13(6):1268-1282.

He S, Chen J, Li J and Sun Y (2014b). Energy-efficient area coverage for intruder detection in sensor networks. *Springer Science & Business Media*.

Jourdan DB and de Weck OL (2004). Multi-objective genetic algorithm for the automated planning of a wireless sensor network to monitor a critical facility. In: *Defense and Security*. International Society for Optics and Photonics, pp. 565-575.

Juan A A, Grasman S E, Caceres-Cruz J, and Bektas T (2014). A simheuristic algorithm for the single-period stochastic inventory-routing problem with stock-outs. *Simulation Modelling Practice and Theory*, 46: 40-52.

Juan A A, Faulin J, Grasman S E, Rabe M, and Figueira G (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2: 62-72.

Karatas M. (2017). A multi objective facility location problem in the presence of variable gradual coverage performance and cooperative cover. *European Journal of Operational Research*, 262: 1040–1051.

Karatas, M. (2018). Optimal Deployment of Heterogeneous Sensor Networks for a Hybrid Point and Barrier Coverage Application. *Computer Networks*. 132 (26), pp. 129-144. DOI: 10.1016/j.comnet.2018.01.001

Karatas M and Onggo BSS (2016). Validating an Integer Non-linear Program Optimization Model of a Wireless Sensor Network Using Agent-based Simulation. In: *Proceedings of the 2016 Winter Simulation Conference*, pp. 1340 – 1351. Los Alamitos, California: IEEE Computer Society Press.

Karatas, M., Yakıcı, E., & Razi, N. (2019). Military Facility Location Problems: A Brief Survey. In Operations Research for Military Organizations (pp. 1-27). IGI Global, DOI: 10.4018/978-1-5225-5513-1.ch001

Kim S.H., & Nelson, B.L. (2006). Selecting the Best System. In: *Elsevier Handboooks in Operations Research and Management Science: Simulation*. Edited by S.G. Henderson & B.L. Nelson. Waltham, MA: Elsevier.

Kumagai, J. 2004. “Life of birds [wireless sensor network for bird study].” Spectrum, 41(4), 42-49.

Kuo, Y. H., Rado, O., Lupia, B., Leung, J. M., & Graham, C. A. (2016). Improving the efficiency of a hospital emergency department: a simulation study with indirectly imputed service-time distributions. Flexible Services and Manufacturing Journal, 28(1-2), 120-147.

Li JH and Yu M (2007). Sensor coverage in wireless ad hoc sensor networks. *International Journal of Sensor Networks*, 2(3 & 4):218-229.

Lin J T, and Chen C M (2015). Simulation optimization approach for hybrid flow shop scheduling problem in semiconductor back-end manufacturing. *Simulation Modelling Practice and Theory*, 51: 100-114.

McCormack R and Cotes G (2015). A simulation model to enable the optimization of ambulance fleet allocation and base station location for increased patient survival. *European Journal of Operational Research*, 247: 294 – 309.

Meketon MS (1987). Optimization in Simulation. In: *Proceedings of the 1987 Winter Simulation Conference*, pp. 58 – 67. Los Alamitos, California: IEEE Computer Society Press.

Morton DP, Pan F and Saeger KJ (2007). Models for nuclear smuggling interdiction. *IIE Transactions*, 39(1):3–14 .

Nugent CD, Hong X, Hallberg J, Finlay D and Synnes K (2008). Assessing the impact of individual sensor reliability within smart living environments. In: *IEEE International Conference on Automation Science and Engineering*, pp. 685-690. IEEE.

Pichitlamken J and Nelson BL (2003). A combined procedure for optimization via simulation. *ACM Transaction on Modeling and Computer Simulation*, 13:155–179.

Rytwinski A and Crowe KA (2010). A simulation-optimization model for selecting the location of fuel-breaks to minimize expected losses from forest fires. *Forest Ecology and Management*, 260: 1 – 11.

Saif A, and Elhedhli S (2016). Cold supply chain design with environmental considerations: A simulation-optimization approach. *European Journal of Operational Research*, 251(1): 274-287.

Saipulla A, Liu B and Wang J (2008). Barrier coverage with airdropped wireless sensors. In: *Military Communications Conference*, pp. 1–7.

Saipulla A, Westphal C, Liu B and Wang J (2009). Barrier coverage of line-based deployed wireless sensor networks. In: *IEEE INFOCOM*, pp. 127–135.

Saipulla A, Westphal C, Liu B and Wang J (2013). Barrier coverage with line-based deployed mobile sensors. *Ad Hoc Networks*, 11(4):1381-1391.

Salmerón J (2012). Deception tactics for network interdiction: A multiobjective approach. *Networks*, 60(1):45-58.

Shen C, Cheng W, Liao X and Peng S (2008). Barrier coverage with mobile sensors. In: *International Symposium on Parallel Architectures, Algorithms, and Networks*, pp. 99-104. IEEE.

Shepherd D and Kumar S (2004). Microsensor Applications. In: *Distributed Sensor Networks*, Sitharama-Iyengar S and Brooks RR (Eds). Clemson University CRC Press, Dec 29, 2004

Ssu KF, Wang WT, Wu FK and Wu TT (2009). K-barrier coverage with a directional sensing model. *International Journal on Smart Sensing and Intelligent Systems*, 2(1):75–93.

Szewczyk R, Osterweil E, Polastre J, Hamilton M, Mainwaring A and Estrin D (2004). Habitat monitoring with sensor networks. *Communications of the ACM*, 47(6), 34-40.

Tao D, Tang S, Zhang H, Mao X and Ma H (2012). Strong barrier coverage in directional sensor networks. *Computer Communications*, 35: 895 – 905.

Wang B. (2010). Sensor Coverage Model. In: Coverage Control in Sensor Networks. *Computer Communications and Networks*. Springer, London, https://doi.org/10.1007/978-1-84996-059-5\_2

Wang B, Chen J, Liu W and Yang LT (2016). Minimum cost placement of bistatic radar sensors for belt barrier coverage. *IEEE Transactions on Computers*, 65(2):577-588.

Wang J, Niu C and Shen R (2009). Priority-based target coverage in directional sensor networks using a genetic algorithm. *Computers and Mathematics with Applications*, 57:1519–1922.

Wang Z, Chen H, Cao Q, Qi H, Wang Z and Wang Q (2017a). Achieving location error tolerant barrier coverage for wireless sensor networks. *Computer Networks*, 112:314-328.

Wang Y, Wu S, Chen Z, Gao X and Chen G (2017b). Coverage Problem with Uncertain Properties in Wireless Sensor Networks: A Survey. *Computer Networks*, 123:200–232.

Washburn AR (1982). On patrolling a channel. *Naval Research Logistics*, 29(4):609-615.

Washburn AR (2015). Barrier Games. *Military Operations Research*, 15(3):31-41.

Werner-Allen G, Lorincz K, Ruiz M, Marcillo O, Johnson J, Lees J and Welsh M (2006). Deploying a wireless sensor network on an active volcano. *Internet Computing*, 10(2):18-25.

Yin PY, Wu TH and Hsu PY (2017). Risk management of wind farm micro-siting using an enhanced genetic algorithm with simulation optimization. *Renewable Energy*, 107:508-521.

Yu L, Wang N and Meng X (2005). Real-time forest fire detection with wireless sensor networks. In: *Proceedings of International Conference on Wireless Communications, Networking and Mobile Computing*, Vol. 2, pp. 1214-1217.

Xu J, Huang E, Chen C H, and Lee L H (2015). Simulation optimization: A review and exploration in the new era of cloud computing and big data. *Asia-Pacific Journal of Operational Research*, 32(03): 1550019.