Path Intrusion Detection Reliability in Wireless Sensor Networks

by

Mohammed Elmorsy

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

The use of Wireless Sensor Networks (WSNs) in various surveillance and monitoring tasks have attracted a wide and growing attention in recent years. The thesis considers a class of intrusion detection problems where an unauthorized person or object aims to traverse part of the geographical area guarded by a WSN. In each problem, the objective is to quantify the likelihood that an intrusion event can be jointly detected and reported to a sink node in a network whose nodes can fail randomly.

The thesis considers three types of intrusion paths: paths across the network with known entry and exit points, paths across the network among a set of entry-exit pairs of points outside the network, and paths from outside the network to an area of interest inside the network. For each of the above types of intrusion paths, the thesis formalizes a corresponding reliability problem that calls for computing the probability that the collaborative work of all nodes in the network succeeds in detecting and reporting an intrusion event.

Our results show that all formalized problems are computationally intractable (#P-hard). Consequently, the thesis focuses on constructing optimized iterative algorithms that utilize special problem structures, known as pathsets and cutsets, for computing exact solutions as well as computing lower and upper bounds. We conclude by presenting some possible future research problems.

Preface

This thesis is the original work done by Mohammed Elmorsy. Publications [18–24] contain original contributions mentioned in Chapters 3 to 7. The work has been done under Professor Ehab Elmallah's supervision. In [24], co-author Professor Hossam AboElfotoh contributed with valuable ideas and insights.

Acknowledgements

First of all, I would like to thank my supervisor, Ehab S. Elmallah, for his guidance during my PhD work. I am very grateful for his sharing of valuable insights that have led to the formulation of the thesis main direction, and his continuous in-depth discussions that have guided me to achieve progress in this direction.

I would like also to thank my thesis examination committee members: Professors Hossam Hassanein, Ioanis Nikolaidis, Janelle Harms, and Zachary Friggstad for reviewing my work and providing valuable insights and suggestions. I would like also to acknowledge financial support from a 2-year AITF (Alberta Innovates Technology Futures) scholarship, and NSERC discovery grants. During my PhD work, I enjoyed working as a member of the network group. I benefited from discussions with M. Shazly, and I. Haque.

Last but not the least, I would like to give special thanks to my lovely wife Salwa Abougamila for her continuous support and encouragement towards achieving my dreams. I would like also to thank my wonderful son Yahia for being our source of joy and happiness. I am very grateful to my parents for their emotional and financial support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Using wireless sensor networks (WSNs) has recently received great attention in many applications such as environmental, medical, military and petroleum applications. A wireless sensor network consists of energy constraint sensor nodes that sense some physical phenomena of importance and send the sensed information to one, or more, sink nodes that perform various control and monitoring functions. The main reasons of using WSNs instead of wired communication in industry are: the high installation cost and maintenance of the wired communication and the demand of continuous growth and safety for industrial environments.

Extensive research work in WSNs has been done over the past decade in many directions such as: energy aware routing, medium access control protocol, topology control, security, privacy and data aggregation.

In harsh industrial environments, however, the communication and sensing capabilities of wireless sensor nodes are subject to random failure due to the presence of obstacles and other interference sources. Analyzing WSNs under node failure effects have not received much attention in existing literature. The research work done in the thesis fills a gap in the current literature by dealing with a number of WSN problems that need to be analyzed in an environment where nodes fail randomly. This work involves formalizing suitable network reliability measures for the problems dealt with, investigating the computational complexity of the formalized problems, as well as investigating the design of efficient algorithms for computing or approximating the resulting reliability measures, and evaluating the performance of the developed algorithms.

In particular, we investigate reliability problems with node failure in the broad class of path exposure and intruder detection problems. Problems in this class concern the ability of a WSN to jointly detect and report to the sink node events of interest. A challenging class of such node events include the detection of unauthorized traversal of a mobile object across a geographic area guarded by a WSN. Chapters 3 to 6 present original new results on several node failure reliability problems in this context.

In the WSN literature, the class of *path exposure* and *intruder detection* problems is closely related to two other classes of problems: localization problems, and tracking problems. Localization problems concern the need for a WSN to self localize the positions of its nodes, as well as localize external targets, or positions of certain monitored events. On the other hand, tracking problems concern detection and timely identification of a target as it moves in the area guarded by the WSN. Each of the above classes of problems have received significant research attention. Despite the existence of many results on problems in these classes, not as much results appear in the context of investigating the network reliability aspects under node failure. In this thesis, we develop methodologies for tackling path exposure and intruder detection reliability problems. The developed methodologies seem to apply to localization and tracking problems. In chapter 2, we give an overview of some existing work on such problems.

The developed methodologies in this thesis have foundations that have been established in the area of investigating combinatorial network reliability problems [15]. In section 1.2, we give an overview of some existing work in this direction. In section 1.3, we present the main aspects of the path exposure and intruder detection problems dealt with in the thesis, and section 1.4 we outline the main contributions of the thesis.

1.2 Network Reliability Problems

In this section, we give an overview of some important existing results in the area of combinatorial network reliability [15]. The majority of existing results in this area have considered various connectivity based and flow based reliability measures. Both types of measures are fundamental in the analysis of various types of transportation and communication networks. Currently, there is a wealth of results on the above measures that forms the basis of our work here. WSN reliability problems, however, typically have joint sensing and connectivity requirements. Thus, tackling WSN reliability problems requires revisiting and extending previously obtained results on connectivity and flow based reliability measures. Not as much work, however, has been done on WSN reliability problems. To provide a background on existing work, we discuss the following important aspects. We follow the presentation in [15, 47] closely.

1.2.1 Basic Definitions and Classes of Reliability Problems

In a connectivity-based (or flow-based) reliability problem, a network is considered operational if it satisfies a certain requirement on node connectivity (respectively, a condition on the amount of flow supported by the network) in an environment where nodes and/or edges of the given network fail. The analysis of such problems utilizes the concept of probabilistic graphs.

A *probabilistic graph* $G = (V, E, p)$ is a graph on a set V of vertices, and a set E of edges (that can be directed or undirected) where each node v and/or edge e operates with a given probability, denoted $p(v)$ (or, $p(e)$), and fails with probability $q(v) = 1 - p(v)$ (respectively, $q(e) = 1 - p(e)$). In such a 2-state model, each component (node or edge) can be either in one of two possible states (operating or failed). The obtained results in the literature concern cases where only nodes can fail, only edges can fail, or both nodes and edges can fail. For convenience, we focus in this section on cases where only edges can fail. In many cases, similar results can be obtained assuming failure of other components.

After an event of random failure, a network whose edges only are subject to failure can be in any one of a possible set of *states*, where a state S is a subset of operating edges after the event (all remaining edges in $E \setminus S$ are failed). Each such a state S arises with probability $Prob(S) = \prod_{e \in S} p(e) \prod_{e \notin S} q(e)$ (where $q(e) = 1 - p(e)$). Thus, if $|E| = m$ then the probabilistic graph G has 2^m states. For a given reliability measure, we denote the set of operating states of the probabilistic graph G by $OP(G)$. The reliability of G is defined as $Rel(G) = \sum_{S \in OP(G)} Prob(S)$.

The 2-terminal reliability problem, denoted Rel_2, is a basic problem in this class. Here, the probabilistic graph G has two distinguished nodes (terminals) denoted s and t. A state S is operating if it provides a path from s to t. In figure 1.1, for example, $S = \{e_1, e_2, e_5\}$ is an operating state. We also need the following definitions:

- A *configuration* of G is a *partial* state where some edges (but not necessarily all) are assigned an operating/failed state; the remaining unassigned edges are *don't care or free* edges.

- A *pathset* is a subset of components (e.g., edges) whose operation yields an operating configuration. A *minpath* is a minimal pathset. For example, in figure 1.1, when each edge in $\{e_1, e_2, e_5\}$ operates, and regardless of the states of the other edges, the source s has an operating path to the destination t. Thus, $\{e_1, e_2, e_5\}$ is a pathset, and in fact, it is a minpath.

- A *cutset* is a subset of components (e.g., edges) whose failure yields a failed configuration even if all unassigned edges operate. A *mincut* is a minimal cutset. For example in figure 1.1, when each edge in $\{e_1, e_4\}$ is failed, and regardless of the states of the other edges, there is no possible path between s and t. Thus, $\{e_1, e_4\}$ is a cutset, and in fact, it is a mincut.

Existing results on connectivity-based reliability problems have considered many cases including the following:

- The probabilistic graph G can be either undirected or directed.

- Component failure can affect edges only, nodes only, or both nodes and edges.

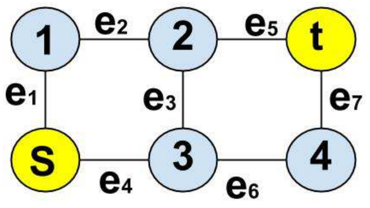


Figure 1.1: An instance of the two terminal reliability problem on terminals s and t

- Component failure probabilities can be either identical for all components or different among the components.
- The reliability measure of interest may consider communication between 2 nodes only, a subset of nodes, or all nodes.

For example, restricting attention to cases where edges only can fail, and edge operational probabilities can be different, gives rise to the following problems:

- **For undirected graphs**: The 2-terminal reliability (Rel_2), the k-terminal reliability (Rel_k) where $k > 2$, and the all-terminal reliability (Rel_{all}) problems
- **For directed graphs**: The 2-terminal connectedness ($CONN_2$), the k-terminal connectedness ($CONN_k$) where $k > 2$, and the all-terminal reachability ($CONN_A$) problems

1.2.2 Reliability Polynomials

When all edge operation probabilities are equal to p, $Rel(G,p)$ can be expressed using the following polynomials:

- Pathset Polynomials:

$$Rel(G,p) = \sum_{i=0}^{m} N_i p^i (1-p)^{m-i} \tag{1.1}$$

  where m is the number of edges in the graph G, p is the operating probability of each edge, and N_i is the number of operating subgraphs with exactly i operating edges, and $m-i$ failed edges (also called i-edge operating subgraph). For example, in figure 1.1, $N_2 = 0$ and $N_3 = 3$.

- Pathset-Complement Polynomials:

$$Rel(G,p) = \sum_{i=0}^{m} F_i p^{m-i}(1-p)^i \tag{1.2}$$

  where F_i is the number of i-edge subgraphs that are complement of pathsets. That is, $F_i = N_{m-i}$. For example, in figure 1.1, $F_4 = 3$ and $F_5 = 0$.

- Cutset Polynomials:

$$Rel(G,p) = 1 - \sum_{i=0}^{m} C_i p^{m-i}(1-p)^i \tag{1.3}$$

  where C_i is the number of i-edge cutsets. A graph is in a failed state when the i edges of a cutset are failed and the remaining $m-i$ edges are operating. Thus, $N_i + C_{m-i} = \binom{m}{i}$.

Some of the uses of reliability polynomials include the following:

- Knowing the coefficients of any polynomial simplifies the following tasks: computing $Rel(G,p)$ for any value p, comparing the reliability of two given networks G_1 and G_2, and analyzing the effect of small changes in p over $Rel(G,p)$.

- Knowing some of the coefficients of any polynomial can be used to derive lower and upper bounds on the unknown coefficients in the same polynomial. Using results in this direction allows deriving lower and upper bounds on the reliability of the entire network.

- Showing that computing certain coefficients to be #P-hard (or NP-hard) can be used to show that computing $Rel(G)$ is #P-hard (respectively, NP-hard).

1.2.3 Computational Complexity

The class of #P-hard (or #P-complete) problems has been introduced in [50] to capture the computational complexity of *counting* problems. All the 6 network reliability problems introduced above have been shown to be #P-complete on arbitrary graphs even in cases of equal operating probabilities. We note the following:

- For radio networks where communication between any two nodes depends on the distance between the nodes, the unit disk graph (UDG) model is useful. UDGs are not arbitrary graphs. Hence, complexity results on UDGs are valuable in the analysis of radio networks.
- Grid graphs where nodes lie on integer coordinates of the infinite grid are all UDGs.
- In [1], the authors show that the Rel_2 on grid graphs is #P-complete.

1.2.4 Exact Algorithms

Given the hardness results of almost all network reliability problems studied in the literature, it is unlikely that exact efficient algorithms exist for solving such problems on arbitrary graphs. Research that aims to find exact solutions has considered the following directions:

1. **Graph simplifications (transformations and reductions):** This approach relies on identifying topological and reliability transformations to convert a given graph G to another graph G' such that: (i) $Rel(G)$ is related to $Rel(G')$, and (ii) G' is either smaller in size or have some special properties that simplifies the computations. Examples of such reductions include removal of irrelevant edges, series reduction, degree-2 reductions, $\Delta - Y$ transformation, and $Y - \Delta$ transformation.

2. **Exact algorithms on special classes of graphs**: Some network reliability problems have been shown to admit polynomial time exact solutions on some special classes of graphs. For example, the following problems are known to have efficient solutions on the following classes of graphs:

   - Rel_2, Rel_k, and Rel_{all} can be solved on trees, series parallel graphs [13], and more generally, the classes of partial $k-$trees [4,5,9], for any $k \geq 1$.
   - Reachability can be solved in polynomial time on directed acyclic graphs.

3. **Exact algorithms on minpaths and mincuts:** For any network reliability problem, exact solutions can be computed from the set of all minpaths or mincuts. This follows since the reliability is the probability that at least one minpath operates (respectively, one minus the probability that all edges in at least one mincut fail). However, the events considered in computing such probabilities are not statistically disjoint, and there is a need to use a suitable inclusion-exclusion formula. Research in this direction has considered achieving efficiency in evaluating such expansion in special classes of problems.

   In another direction, research efforts considered the possibility of obtaining exact solutions in time that is polynomial in the number of minpaths and mincuts. For example, [43] has shown that Rel_2 can be solved in time which is polynomial in the number of mincuts. In contrast, it has been shown that computing $CONN_2$ is #P-hard even a list of all minpaths is given as part of the input.

1.2.5 Bounding Techniques

Given the computational intractability of various network reliability problems, many research results have focused on obtaining lower and upper bounds on the exact solution. The following describes some existing approaches:

- **Bounds from reliability polynomials:** As mentioned above, in cases where all edges have equal operating probability p, the exact reliability can be expressed as a polynomial in p. Thus, a lower (or upper) bound on the exact solution can be obtained by establishing lower (respectively, upper) bound on each coefficient in such a polynomial. In [15], several results in this direction are presented.

- **Bounds from subgraphs (or supergraphs):** In cases where edges fail but nodes are reliable, one can obtain bounds on a given network reliability problem by the following means:

1. Approximating the structure of a given graph G by a subgraph G' to get a lower bound (or, a supergraph G'' to get an upper bound).
2. Decomposing a graph G into a set of edge-disjoint subgraphs $G_1, G_2, \cdots, G_r$.

In point (1), we seek a subgraph G' (or a supergraph G'') that has a special structure that enables exact solution of the given reliability problem.

In point (2), we compute a lower bound on $Rel(G)$ using the probability that at least one of the r edge-disjoint subgraphs is operating.

- **Bounds from pathsets and cutsets:** Given a set of pathsets $\{P_1, \cdots, P_r\}$, one can compute a lower bound on the exact reliability by computing the probability that at least one pathset is operating. Similarly, given a set of cutsets $\{C_1, \cdots, C_r\}$, one can compute an upper bound on the exact reliability by computing the probability that at least one edge in each cutset is operating. Existing results in this direction obtain bounds from pathsets (or cutsets) that are either *edge-disjoint*, or have a special property, called the *consecutive* [47,49] set property.

  In addition, one can obtain bounds from operating (or failed) configurations that are statistically disjoint (abbreviated s-disjoint): Two configurations are s-disjoint if and only if there exists at least one edge that is assigned two different states in the two configurations.

1.3 WSNs Detection Reliability Problems

In this section, we describe three types of WSN detection problems that are investigated in the thesis.

1. **The Path Exposure Problem:** The notion of path exposure in WSNs is a fundamental notion that has been investigated in the early work of [39]. Informally, exposure is a measure of how well an object moving on an arbitrary path can be observed by the sensor network over a period of time [39]. Factors that determine path exposure in a WSN include: the amount of emitted

energy from the target that is received by the sensor nodes (including noise effects), the target moving speed, the sampling rate adopted by the sensor nodes, and the data fusion method that determines target detection from the sensed data.

In [14], target detection probability is analyzed under two data fusion models. In *value fusion*, a center node gathers energy measurements from all sensors and use the aggregate information to reach a decision. In *decision fusion*, each sensor processes its own sensed data and reach an individual decision. Such decisions are forwarded to a sink node that decides whether or not an object has moved along some path in the network. In [38], the authors have investigated the best and worst exposed paths (called maximal support path and maximal breach path, respectively).

Given the importance of the path exposure problem, we conclude that it is worthwhile quantifying the network's ability to successfully detect and report an intrusion through a given network in an environment where nodes fail independently of each other.

2. **Breach Path Detection Problem:** In the class of path exposure problems discussed above, the path along which a target moves across the network is either given as part of the input, or we seek to find a path of minimum or maximum exposure. More generally, one may only know the start and end points, outside the area guarded by the WSN, of the path across the network used by an intruder. In this more general class of problems, the intruder is free to use any possible path across the network whose start and end points are prescribed. We want to analyze the likelihood that the WSN will be able to jointly detect and report intrusion.

3. **Breach Path to Target Area Reliability Problem:** In the previous class of problems, the intruder's trajectory lies across the network. In this class of problems, the intruder seeks to enter the network from outside to reach a particular area of interest inside the network. As above, we want to analyze the likelihood that the WSN will be able to jointly detect and report intrusion.

1.4 Thesis Contributions and Organization

In this section, we summarize the main contributions of the thesis. The contributions concern the following four path intrusion network reliability problems. In each problem, we aim at quantifying the likelihood that the collaborative work of all nodes in the network succeeds in jointly detecting and reporting a path traversal across the network.

1. In Chapter 3, we formulate *a path exposure reliability problem*, denoted $EXPO$, where an intruder path is given, and the sensing and communication modules of each node can fail independently of each other. The network is operational if it succeeds to detect and report to the sink the given path.

2. In Chapter 4, we formulate *a breach path detection reliability problem*, denoted BPDREL, where a set of entry-exit sides of possible intrusion paths are given. The network is operational if it succeeds in detecting and reporting intrusion along any possible path.

3. In Chapter 5, we formulate *a directional breach path detection reliability problem*, denoted $\mathrm{DIR\text{-}BPDREL}$. This problem extends the $BPDREL$ problem to networks with nodes equipped with directional sensing and communication modules.

4. In Chapter 6, we formulate *a breach path to target area reliability problem*, denoted $\mathrm{BPTA\text{-}REL}$. In this problem, an intruder aims to reach an area of interest inside the network.

We show that each of the above problems is #P-hard. This motivates further work to find effective algorithms to handle the problems. To this end, we adopt a framework that yields lower bounds (LBs) and upper bounds (UBs) on exact solutions. The framework is based on using an iterative algorithm that utilizes pathsets (and cutsets) to derive LBs (respectively, UBs). The algorithm can produce exact solutions if allowed to execute for a sufficient number of iterations.

A core step in the iterative algorithm is to extend a given input configuration to a pathset (or a cutset). Recall that, an input configuration assigns the operating

Table 1.1: A summary of contributions

Problem	**Complexity**	**Node Reliability Model**	**Extension to a Pathset** E2P		**Extension to a Cutset** E2C	
			Complexity	Ability to recognize all extensible inputs	Complexity	Ability to recognize all extensible inputs
$EXPO$	#P-hard	3-state		Yes		Yes
BPDREL	#P-hard	2-state	NP-c	Yes	P	Exact
DIR-BPDREL	#P-hard	3-state	NP-c	Yes		Yes
BPTA-REL	#P-hard	2-state		Yes		Yes

(or failed) state to some nodes, and keeps the remaining unassigned nodes as free nodes. We note that each of the above mentioned four WSN reliability problems, one can decide in polynomial time whether a given configuration is a pathset, or a cutset. Therefore, a simple algorithm to extend an input configuration to a pathset (if one exists) may proceed by assigning the operating state to a sufficient number of free nodes so as to obtain a pathset. Likewise, extending an input configuration to a cutset (if one exists) proceeds by failing a sufficient number of free nodes.

In each core step, it is desirable to compute an extension to a pathset (or a cutset) that has the highest occurrence probability. The use of an extension that has a high occurrence probability increases the benefit of each iteration at the cost of increasing the running time per iteration. For each of the four reliability problems, we formulate a corresponding optimal extension to a pathset (E2P) problem, and an optimal extension to a cutset (E2C) problem. And, we devise algorithms for each of the resulting E2P and E2C problems (a total of 8 problems). The following table summarizes our contributions.

In the table, a 2-state node reliability model refers to a model where each node can either be operating or failed. Here, a node fails if either its sensing module fails or its communication module fails. In a 3-state node reliability model, a node can either be in a communicate and sense state, a communicate but not sense state, or a failed state. Here, a failed node can not communicate with its neighbors.

1.5 Concluding Remarks

In this chapter, we have motivated research work on developing efficient algorithms for assessing the reliability of WSNs in an environment where nodes fail randomly.

We have included a review of some of the main research directions considered in the literature for analyzing connectivity-based reliability problems. We have also presented the main problems investigated in the thesis and outlined the thesis contributions.

Chapter 2

Literature Review

Our focus in this thesis is on the class of path exposure and intrusion detection reliability problems. In the WSN literature, the three classes of *localization*, intrusion *detection*, and intrusion *tracking* are closely related to each other. For a broad understanding of the relation of the problems dealt with in the thesis with other problems, we review in this chapter some results in all three classes. we remark that our presentation in chapters 3 to 6 are intended to be self-contained that does not depend on the detailed information provided in this chapter.

2.1 Localization in WSNs

The operation of many WSNs require the ability to determine the location of the monitored targets or events. However, generally, in a randomly deployed WSN, nodes are not initially aware of their locations in either absolute or relative coordinates. The purpose of a localization algorithm is to enable the network to self localize its nodes over time. A vast amount of results appear on localization algorithms (see, e.g., the surveys in [11,29]). Existing approaches refer to the following concepts:

- The nodes in a WSN are assumed to include anchor (or landmark) nodes that know their position either in absolute coordinates (e.g., using Global Positioning System(GPS)), or relative to each other. Other nodes are initially not aware of their positions. Such nodes are sometimes referred to as *unknown* nodes.

- In *range-based* localization methods, a node estimates its distance and/or angle to a collection of anchor nodes. To achieve this purpose, a node can examine physical quantities such as received signal strength indicator (RSSI), time of arrival (TOA), time difference of arrival (TDOA), angle of arrival (AOA), etc. (chapter 1 in [6], [11], and chapter 11 in [12])

- In *range-free* localization, only connectivity and routing information are used to estimate distances between landmarks and other nodes. No direct measurement of distances or angles is used. Therefore, range-free localization methods do not require any additional equipment in the localization process. On the other hand, range-free localization methods generally have lower accuracy than range-based localization methods.

- In *centralized* localization approaches, distance and/or angle information collected by all nodes are transmitted to a central node that computes the position of each node. Alternatively, in *distributed* localization approaches, each node collects distance and/or angle information from a sufficient number of neighbouring anchor nodes and executes an algorithm to infer location information (e.g., *trilateration*, maximum likelihood *multilateraion*, or *triangulation* algorithms) [11, 12].

The above concepts have been developed and applied largely in the context of WSNs where all nodes (anchor and ordinary nodes) are stationary. Motivated by applications where some of these nodes are mobile, recent research work has developed localization algorithms to deal with the following cases (see, e.g., the survey in [29]):

- WSNs with static landmarks and mobile nodes (5 papers introduced in [29]): In [29], the authors identify two classes of algorithms where ordinary nodes are mobile.

  - In the class of *historical information* localization algorithms, a predictive location estimation technique is used to estimate future locations of mobile nodes based on their previous locations. The use of predictive methods are necessary where no enough landmark nodes exist, or when noise affects the localization process.

  - In the class of *cluster-based* localization algorithms, the deployment area is divided into clusters where each cluster contains some landmarks. The landmarks associated with each cluster are responsible for localizing unknown nodes inside their associated cluster only.

- WSNs with mobile landmarks and static nodes (9 papers are introduced in [29]): One objective of localization algorithms in such networks is to minimize the number of mobile landmarks needed in the localization process. The mobile landmarks are assumed to have enough power to move through the entire network. Therefore, each unknown node can estimate its location by connecting to the mobile landmarks. The classification introduced in [29] identify the following classes.

  - In the class of *geometric* localization algorithms, some geometric properties are used to estimate the locations of the unknown nodes.
  - In the class of *path planning* algorithms, the general objective is to optimize the path taken by the mobile landmark nodes with respect to a given objective function.

Other localization algorithms for the class of WSNs with mobile landmarks and mobile nodes are surveyed in [29].

2.2 Detection in WSNs

A core operation of any WSN is to detect events of interest and report such events to the sink nodes. To determine the location of a target, it is assumed that each node knows its location (e.g., using one of the localization algorithms discussed in the previous section), and that the target or event of interest is detected by a sufficient number of nodes. Work in the literature on detection is diversified since it typically considers many factors such as:

- The physical layer models and parameters used in describing the quality of node sensing (e.g., the node's sensing sensitivity function, sensor sampling rate, and target speed)
- The exact method of fusing the collected sensed data to decide on target presence, or event occurrence
- The type of target or event under consideration (e.g., a discrete object crossing the boundary, or a continuous object like a spreading fire)

In this section, we present a few selected results that are representatives of the following directions:

- Detection using directional sensor nodes
- Detection in presence of obstacles
- Detection enhancement in randomly deployed networks

2.2.1 Detection Using Directional Sensor Nodes

In [37], the authors have considered path exposure problems in WSNs employing directional sensors. As described below, the concept of path exposure used in the paper quantifies the cumulative amount of energy sensed from all points on a given path. Given two end points, the paper first develops a quantitative model to compute path exposure for a given path between two given end points. Subsequently, the paper devises a discretized method to find an approximation of a path with minimum exposure. The main idea of the devised method has been previously used in [39] for WSNs with omnidirectional sensors. We briefly introduce some key notation and quantities in [37] used in the analysis.

For a sensor node s_i and a target point T, we have the following quantities:

- $d(L_i, T)$: the distance between the location of s_i (denoted L_i) and the location of T
- $\phi(\overrightarrow{V_i}, \overrightarrow{L_iT})$: the angle between the direction of sensor s_i (i.e., the direction of maximum reception), and the line segment between L_i and T

- $\omega(s_i, T)$: the sensitivity of sensor s_i to a target at point T
- μ, β, and γ: parameters used in the next equation to model $\omega(s_i, T)$

Using the above notation, the authors adopt the following **sensitivity** model where attenuation occurs if distance increases and/or the angle ϕ increases:

$$\omega(s_i, T) = \frac{\mu(cos[\frac{\phi(\overrightarrow{V_i}, \overrightarrow{L_iT})}{2}])^{\beta}}{[d(L_i, T)]^{\gamma}} \tag{2.1}$$

The exposure of a horizontal line segment $e_{a,b} = ((x_a, y_{common}), (x_b, y_{common}))$ is computed as follows (similar definitions apply to a vertical line segment $e_{a,b}$):

1. For a single node s_i, the exposure of the line segment $e_{a,b}$ to sensor s_i is given by $E_i(e_{a,b}) = \int_{x_a}^{x_b} \omega(s_i, p)dx$, where $p = (x, y_{common})$ is any point in the line segment.

2. When the *all-sensor* field intensity function is used in a WSN with S nodes, the exposure is given by $E_i(e_{a,b}) = \int_{x_a}^{x_b} \sum_{s_i \in S} \omega(s_i, p)dx$

3. When the *maximum* field intensity function is used in a WSN with S nodes, the exposure is given by $E_i(e_{a,b}) = \int_{x_a}^{x_b} max_{s_i \in S}(\omega(s_i, p))dx$

To approximate the problem of finding a path with minimum exposure, the authors confine the search to paths composed of horizontal and vertical edges of a grid superimposed on the WSN's area. Moreover, the approximation neglects the contribution of a sensor s_i to point p if $\omega(s_i, p) < \epsilon$, for some desired threshold ϵ. A path is between two end points corresponding to the intruder's entry and exit points. The methodology involves the following steps:

1. Using either the *all-sensor* field intensity function (point 2 above), or the *maximum* field intensity function (point 3 above), compute the exposure of every edge in the grid

2. Using a shortest path algorithm, compute a path through the grid with a minimum cumulative exposure of all of its edges. The computed path is the desired approximation to the solution of the problem.

2.2.2 Detection in Presence of Obstacles

In [41], the authors have considered a WSN deployed in a rough terrain with obstacles that can obstruct sensing. The network guards against unauthorized traversal through the monitored region. The following assumptions and notation are used to define the problem in [41]:

- The deployment area is modelled by a long narrow rectangular grid of dimensions $N \times M$, where $N > M$. The grid serves as a segmentation of the continuous area monitored by the WSN (so, sensor nodes do not necessarily lie on grid points).

- One long side of the grid, say the top side, separates a secure area in the field from the area monitored by the WSN (i.e., the grid area). The other long side, say the bottom side, separates an insecure area from the grid area. The intruder aims at crossing the grid from the insecure area to the secure area.

- Sensing through an obstacle is impossible. Thus, a sensor can detect a target point only if there is an unobstructed line-of-sight between the sensor and the point. Grid points inside an obstacle are therefore deleted from the grid. We denote by G the partial grid obtained by deleting grid points that lie inside each obstacle.

- Each grid point p has an associated exposure level, denoted $I(p)$, corresponding to the probability that the overall WSN can detect a target at that point.

- For a given path P of grid points, the detection probability of P is the maximum exposure level of a grid point on P. We denote such a value by $I_{max}(P)$ (this notation is not in [41]).

- The weakest breach path, denoted P^*, is defined as a path crossing the network from the insecure area to the secure area with the minimum possible detection probability.

- The deployment quality measure (DQM) is the value $I_{max}(P^*)$.

The problem defined in [41] is to compute the DQM for a given WSN. The work in [41] defines a graph, denoted H, that is typically smaller than the partial grid G. The authors utilize such a graph H to solve the DQM problem in polynomial time. The main solution strategy, however, can be explained using the partial grid G instead of using the smaller graph H. In more detail, the solution strategy using the grid can be explained as follows:

- Construct a graph $G_{s,d}$ by adding two new nodes, denoted s and d, where s (respectively, d) is connected to every grid point on the secure (respectively, insecure) side of the grid. Thus, every $(s,d)-$path across the WSN is a breach path.

- Construct a list L of exposure levels of the grid nodes sorted from highest to lowest. Search L for the smallest value of x such that if one omits from $G_{s,d}$ all grid points whose exposure level is above x then $G_{s,d}$ continues to have $(s,d)-$path. Set $DQM = x$

The work in [41] considers the following additional details:

- Constructing a graph H that is typically smaller than the partial grid G utilizes the following observations.
  - A path P that has a small detection probability $I_{max}(P)$ follows grid points that are either (a) most distant from sensor nodes, or (b) provide a passage around an obstacle. Type (a) points correspond to grid points of G whose exposure satisfy a local minima property. By connecting such grid points, one obtains a set of contours that include the weakest breach path. Fig. 2.1 shows an example of such contours that are used to obtain possible intruder's breach paths.
  - The authors remark that the use of Voroni diagrams are not suitable to identify such contours due to the existence of obstacles. Instead, a segmentation method (due to [53]) that borrows ideas from image segmentation (and segmentation of topological maps) is used in [41].

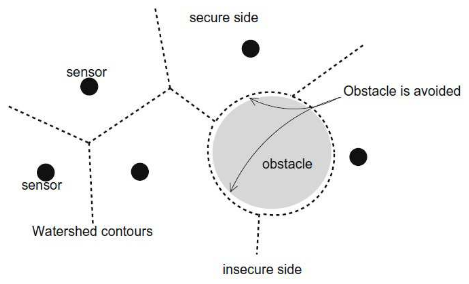


Figure 2.1: An example of possible intruder breach paths

- The paper utilizes a sensor sensitivity model where the probability of detecting a target by a sensor decays exponentially with the sensor-to-target distance.

- The paper experiments with 4 different functions to determine the exposure level $I(p)$ for any point p in the field. The four functions differ in the use of the number of sensor nodes participating in the function, and the sensitivity of each of the involved sensors.

2.2.3 Detection Enhancement in Randomly Deployed Networks

In [31], the authors consider random deployment of a WSN to protect a circular area, denoted A, against unauthorized traversal. The following assumptions and notation are used to define the problem:

- The sink node lies at the center of the circle A at $(x, y)-$coordinates $(0, 0)$.

- The quality of coverage requirements to be achieved by the devised deployment scheme are specified by the following input values:

  – P_s^*: a sensing coverage probability requirement for a border area of circle A. This requirement translates to (a) identifying a circular ring area on the perimeter of circle A, and (b) ensuring that the probability that each point in this identified ring $\geq P_s^*$. The sensing coverage requirement aims at controlling the distance that the intruder traverses within A before detection.

  – P_c^*: a connectivity probability for the communication graph induced by all sensor nodes deployed in the circle A. Achieving higher connectivity of such a graph provides better protection, especially to the central area of A.

The paper introduces a random network deployment scheme that allows for optimizing the number of deployed nodes while achieving the required P_s^* and P_c^* levels. The scheme utilizes the following two random distributions.

- Uniform random deployment using a 2-dimentional **Poisson** point process of density λ: A randomly deployed network using this distribution over a region D of area $|D|$ deploys an average of $\alpha = \lambda|D|$ nodes in the area. Thus, the number of nodes deployed in the area, denoted $N(D)$, obeys the Poisson distribution: $P(N(D) = k) = \frac{\alpha^k e^{-\alpha}}{k!}$. The authors remark that although a uniform random deployment over the circle A can be tuned to satisfy the required coverage and connectivity levels, it is not particularly efficient if certain areas within A (e.g., the central area) needs extra coverage.

- Deployment using a 2-dimentional **Gaussian** distribution with mean point (x_i, y_i) and variances (σ_x, σ_y). This distribution provides a means of providing better protection to the area in the vicinity of the mean point (x_i, y_i). In the paper, the central area is assumed to require more protection, hence, the paper sets $(x_i, y_i) = (0, 0)$.

The main idea of the paper is to achieve the coverage, and connectivity requirements (i.e., P_s^* and P_c^*) by utilizing a hybrid deployment scheme, called *Gaussian-Ring* scheme, where a Gaussian distribution is used to cover the whole area A, and a Poisson distribution is used to add more nodes to the identified border ring. In more detail, if r_s and r_c denote the sensing radius, and the communication radius of each sensor node, respectively, and R denotes the radius of circle A, then the analysis decomposes A to $k = \lceil R/r_c \rceil$ rings, where each ring has width $\leq r_c$. The innermost ring (with index $i = 1$) is a circle of radius r_c. The outermost ring (with index k) may have width $\leq r_c$. The analysis utilizes the following two important results.

- In [36], the authors obtain asymptotic results on the sensing coverage of an area by sensors deployed according to a Poisson distribution with density λ. Results on deployment in two types of areas are reported: large areas, and long rectangular strips of width W (W is the smaller dimension of the strip). Both the Boolean sensing model and a more general sensing model are considered. In each case, sensors of sensing radius r_s are deployed. In cases of the Boolean sensing model, and either a large area, or a long strip area, where $W >> r_s$, the authors show that the probability that a point p is covered by at least one sensor node is

$$f_a = 1 - e^{-\lambda\pi r_s^2} \tag{2.2}$$

- In [7], the author obtains asymptotic results on the connectivity of sensors deployed according to a Poisson distribution with density λ. The author shows that if n nodes are deployed, and the communication radius of each node is r_c then the probability that no node is isolated (i.e., the minimum degree $d_{min} > 0$) is given by

$$P(d_{min} > 0) = (1 - e^{-\lambda\pi r_c^2})^n \tag{2.3}$$

  Thus, r_c achieves a desired $P(d_{min} > 0) = p$ value if

$$r_c \geq \sqrt{\frac{-ln(1 - p^{1/n})}{\lambda\pi}} \tag{2.4}$$

  The author combines the above result with a result of [42] that shows that if n is high enough then as r_c increases, the resulting random geometric graph becomes $k-$connected at the moment it achieves a minimum degree of k. The combined result is that the bound given by equation (2.3) is tight to achieve $1-$connectedness.

We now outline the approach devised in [31] to optimize the number of nodes deployed by the Gaussian-ring scheme. Below, N_g and N_u denote the average number of nodes deployed by the Gaussian distribution and the uniform (i.e., Poisson) distribution, respectively. So, we wish to minimize $N_g + N_u$. In addition, we assume that P_c^*, P_s^*, r_c, r_s, and $\sigma_x = \sigma_y = \sigma$ are given (note: another optimization problem arises if σ is not given as part of the input).

1. To find the minimum acceptable number N_g of nodes, the paper applies the following steps.

   (a) Find $E(N_{g,k})$, the expected number of Gaussian nodes that fall in the outermost ring k

   (b) Denote the area of ring k by $|A_k|$, compute the density $\lambda = E(N_g, k)/|A_k|$ of Gaussian nodes that fall in ring k

   (c) Assuming that the Gaussian nodes in ring k form a Poisson distribution with density λ, the paper uses equation (2.3) to find N_g that gives the desired connectivity probability P_c^* in ring k

   (d) Since inner rings have higher density of Gaussian distributed nodes, the connectivity probability P_c^* is also satisfied in each inner ring.

2. To find the minimum acceptable number N_u of nodes deployed in ring k to achieve the desired coverage probability P_s^*, the paper uses equation (2.2) to find the minimum density λ required to achieve P_s^* in area $|A_k|$, and then sets $N_u = \lambda|A_k|$.

Similar to points 1.c and 1.d, connectivity of the overall graph with probability P_c^* relies on having a subset of Poisson distributed points with density satisfying equation (2.3) where $P(d_{min} > 0) = P_c^*$.

2.3 Tracking in WSNs

WSNs are also used to track movement of both discrete and continuous objects over time. Discrete objects have fixed shape and size whereas continuous objects change their size and shape over time (e.g., wild fire, and bio-chemical material). In general, continuous objects tend to diffuse, and can split into multiple smaller continuous objects. Tracking involves periodic detection, localization, and reporting of such objects. Many research results exist in this area (cf., the surveys in [8,17,40]). The results span both networking algorithms and protocols, and signal processing algorithms (e.g., [35]). In this section, we briefly mention some of the techniques, classifications, and performance metrics that have received attention in the networking literature.

Tracking algorithms are sometimes considered in conjunction with techniques from the following areas.

- **Data aggregation** techniques that aim at reducing and compressing data exchanged during the tracking process so as to reduce energy consumption (see, e.g., [25–27])
- **Data fusion** techniques that aim at obtaining inference about the collected data and reducing error by eliminating noisy sensed measurements (see, e.g., the survey in [48])
- **Mobility prediction** techniques that aim at predicting the movement direction and speed of the target object over a short future period of time (see, e.g., [54])

In [8,40], the authors classify many of the existing results in the tracking literature along the following dimensions. (To elaborate on the classification further, we include the number of papers cited under each of the identified classes.)

- **Classification with respect to the network structure:** Various network topologies are considered for usage in tracking tasks. Typical examples include tree-based networks (8 papers cited in [8]), cluster-based networks (14 papers cited in [8]), and leader-based networks (4 papers cited in [40]).
- **Classification with respect to the number of objects tracked:** Results on both single target tracking (4 papers in [40]), and multi-target tacking (8 papers in [40]) appear in the litrature.
- **Classification with respect to the type of object tracked:** Work on both discrete object tracking (12 papers in [40]), and continuous object tracking (2 papers in [40]) exist in the literature.

Many performance metrics are used in analyzing the proposed tracking algorithms and protocols. Of the available metrics, we mention the following:

- Latency until the network estimates the target location since its first presence in the monitored area
- Latency of reporting to the sink node the estimated target location
- Energy consumed by the active nodes required to track the object

2.4 Concluding Remarks

In this chapter, we have discussed three fundamental categories of WSN tasks: localization, detection, and tracking tasks. The negative effect of node failure in each of the above tasks can be quantified in a measurable way. For example, node failure may result in

- a large fraction of unlocalized nodes in a localization task,
- a large rate of false negatives in a target detection task,
- a high probability of losing the itinerary of an object in a tracking task.

In the next chapters, we present our obtained results on analyzing a number of target detection problems in an environment where nodes fail independently of each other.

Chapter 3

Path Exposure Problem in WSNs

In this chapter, we consider wireless sensor networks for surveillance applications where a node's ability to detect and report intrusion is described probabilistically. In addition, intruders traversing an area may probabilistically disrupt sensors by spreading jamming devices. We formalize a problem called the path exposure (EXPO) problem that quantifies the likelihood that a WSN can detect and report intrusion events. Our main contributions in this chapter are as follows:

1. We show that the EXPO problem is #P-hard even on grid networks.
2. We introduce an algorithm for computing lower and upper bounds on the solution for networks with arbitrary topology where the sink location is unconstrained. The algorithm extends the idea of factoring algorithm presented in [15] to work on networks with 3-state nodes.
3. We present simulation results analyzing various aspects and applications of the algorithm. One application concerns an intruder traversing a path across the network and using jamming devices.

Some of the results in this chapter appear in [24].

3.1 System Model

3.1.1 Network Connectivity

Path exposure problems in the literature are typically discussed using a graph model for the underlying WSN. We denote such graph by $G = (V \bigcup \{s\}, E)$, where s is a sink node that performs various network control and data collection tasks, and E a set of communication links. For simplicity of presenting the algorithm, we assume that links are bidirectional. The algorithm, however, can be adapted to use directional links.

Due to operation in harsh environments, possible energy depletion, and effect of intentional jamming, a node x is assumed to succeed in communicating with its neighbours with a probability, denoted $p_{relay}(x)$.

3.1.2 Probabilistic Sensing Model

Many existing work on target detection and path exposure relies on the following relations. Given a sensor node x and a static target at a point p that is d units away from x, the average power emitted by the target and received by x fades as d increases according to some path loss model.

To detect a target, a sensor node performs a number of detection attempts. In each attempt, the sensor accumulates the received energy over some period of time that depends on the sensor technology. By considering the average received energy and noise power in a time period Δt one can determine the probability $p_{sense}(x, p)$ of detecting the target point.

A target moving along a path P that crosses x's sensing region may be detected by a number of detection attempts where each attempt is associated with a segment traversed by the mobile target. The probability of detecting the target in any of these segments is denoted $p_{sense}(x, P)$.

Thus, $p_{sense}(x, P)$ is likely to decrease as the target moves fast since the number of probing segments of P may decrease, and/or the frequency of detection attempts may decrease. In both cases, the received energy collected from the target is likely to decrease. In our model, the intruder may follow a path of arbitrary shape (e.g., a wavy curve). We simplify the notation $p_{sense}(x, P)$ to $p_{sense}(x)$.

3.1.3 Intruder Jamming

In addition, we assume that an intruder can spread a number of jamming devices to hamper the network's communication and/or sensing ability. Each jamming device has a radius of effect, denoted R_{jam}. In our model, its effect is captured by reducing $p_{relay}(x)$ and/or $p_{sense}(x)$ of each affected node x.

3.2 Problem Formulation and Remarks

We now define the path exposure on probabilistic WSNs problem. To focus on the combinatorial aspects of the problem, our definition assumes that for any node x, the sensing probability $p_{sense}(x)$ and the relaying probability $p_{relay}(x)$, have been precomputed from physical problem parameters such as the specified intruder path, the intruder's speed, the frequency of detecting attempts performed by sensor nodes, and so on. We also assume that nodes fail to sense events and relay data independently of each other. The sink is assumed to be perfectly reliable, but it does not participate in sensing activity.

Definition (the EXPO Problem): Given a WSN $G = (V, \{s\}, E)$ where s is the sink node, a path P, and integer $k_{req} \geq 1$, where each node $x \in V$ has a probability $p_{sense}(x)$ of detecting the path P, and a probability $p_{relay}(x)$ of successfully forwarding data to its neighbours, compute the probability $Expo(G, P, k_{req})$ that P can be detected by k_{req} or more sensor nodes connected to the sink node. ■

Throughout the chapter, we also use the abbreviated notation $Expo(G)$, $Expo(G, P)$, and $Expo(G, k_{req})$ to emphasize important parameters in a specific context.

3.2.1 Complexity Analysis

Theorem 3.1 *The EXPO problem is #P-hard.*

Proof. In [1], the authors have shown that computing the two-terminal reliability of a wireless partial grid network ($2REL$) is #P-Complete even when the problem is restricted to partial grid networks of equal operating probabilities and equal communication ranges. An instance (G, s, t) of the $2REL$ problem is specified by a probabilistic graph $G = (V, E)$ on a set V of nodes, a set E of links, and two distinguished nodes s and t. The problem asks for the computing the probability, denoted $Rel(G, s, t)$, that at least one operating path between s and t exists in the network.

To show that $EXPO$ is #P-hard, we reduce in polynomial time any given instance (G, s, t) of the wireless $2REL$ problem to an instance (G, P, k_{req}) of the $EXPO$ problem. The reduction works as follows:

1. For each node $x \in V$ in the constructed $EXPO$ instance, set $p_{relay}(x) = p(x)$ ($p(x)$ is the operating probability of node x in the $2REL$ instance).

2. Set t as the sink node in the $EXPO$ instance.

3. In the $EXPO$ instance, set $p_{sense}(s) = 1$, and set p_{sense} of other nodes to zero.

4. Fix a path P in the $EXPO$ instance to be any path that can be sensed by node s, and set $k_{req} = 1$.

The proof follows since $Rel(G, s, t) = Expo(G, P, k_{req})$. ■

3.3 States, Configurations, Pathsets, and Cutsets

In this section, we define the concepts of node states, network states, network configurations, pathsets, and cutsets. These concepts are needed to present our main algorithm.

3.3.1 Node States

A non-sink node x can be in any one of 3 possible states. State rs where x can perform both relaying and sensing. So, $p_{rs}(x) = p_{relay}(x) \cdot p_{sense}(x)$. State rns where x can perform relaying but not sensing. So, $p_{rns}(x) = p_{relay}(x) \cdot (1 - p_{sense}(x))$. State $fail$ where x can not perform relaying. So, $p_{fail}(x) = 1 - p_{relay}(x)$.

3.3.2 Network States

When each node in G assumes one of the 3 possible states, we obtain a *state* of the network G (thus, G has $3^{|V|}$ states). Any such state T is defined by 3 disjoint subsets of nodes: T_{rs} (nodes in state rs), T_{rns} (nodes in state rns), and T_{fail} (failed nodes), where $V = T_{rs} \cup T_{rns} \cup T_{fail}$.

If $k_{req} \geq 1$ is the minimum number of sensor nodes required to detect intrusion then a state T is operating only if $|T_{rs}| \geq k$, and at least k of the sensors in T_{rs} reach the sink. The probability that state T arises is

$$\Pr(T) = \prod_{v \in T_{rs}} p_{rs}(v) \prod_{v \in T_{rns}} p_{rns}(v) \prod_{v \in T_{fail}} p_{fail}(v).$$

And, the required exposure function $Expo(G, k_{req}) = \sum \Pr(T : T$ is an operating state$)$.

The above expression can be used to evaluate the exposure function by exhaustively generating all distinct states of the network and summing over the set of operational states. For a network with $|V| = n$ nodes, such algorithm requires $O(3^n)$ time. Hence, its practical use is limited to networks of small sizes.

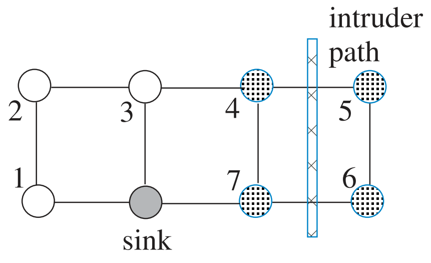


Figure 3.1: An instance of the EXPO problem

Example. Fig. 3.1 illustrates an instance of the EXPO problem where the network G has $|V| = 7$ nodes. Thus, G has 3^7 states. We assume that only nodes represented with dotted circles can sense the intrusion path, other nodes can only relay data. Consider the network state T where $T_{rs} = \{5\}$, $T_{rns} = \{3, 4\}$, and the remaining non-sink nodes are failed. If $k_{req} = 1$ then T is an operating state. Else (if $k_{req} \geq 2$), then T is a failed state. ■

3.3.3 Network Configurations

A network *configuration* assigns a state to some (but not necessarily all) nodes. For a node v and a state $s_v \in \{rs, rns, fail\}$, we write (v, s_v) to mean that node v is in state s_v. We also find it convenient to write a probability $p_{s_v}(v)$ as $p(v, s_v)$. For example, in the network of Fig. 3.1, assume that $k_{req} = 1$. Thus, configuration $C = ((5, rs), (4, rns), (3, rns))$ is operating regardless of the state of the remaining nodes in V.

Two configurations C_1 and C_2 are statistically disjoint (**s-disjoint**) iff at least one node that occurs in both configurations is assigned different states in the configurations. Thus, $\Pr(C_1 \cup C_2) = \Pr(C_1) + \Pr(C_2)$.

3.3.4 Pathsets and Cutsets

A *pathset* is an operating configuration. In Fig. 3.1, e.g., $C = ((5, rs), (4, rns), (3, rns))$ is a pathset assuming $k_{req} = 1$. A *cutset* is a configuration that can not be extended to a pathset. In Fig. 3.1, e.g., $C = ((3, fail), (7, fail))$ is a cutset for any $k_{req} \geq 1$.

The main idea used in our devised algorithm to obtain a lower bound on the exposure function is to compute a number of pairwise s-disjoint pathsets, say $P_1, P_2, \cdots, P_r$, and use the inequality $Expo(G, k_{req}) \geq \sum_{i=1}^{r} \Pr(P_i)$ where the RHS is the computed lower bound. To compute an upper bound, our algorithm computes a number of pairwise s-disjoint cutsets, say $U_1, U_2, \cdots, U_r$, and use the inequality $Expo(G, k_{req}) \leq 1 - \sum_{i=1}^{r} \Pr(U_i)$ where the RHS is the computed upper bound.

Similar inequalities have been used by various authors in the literature. In [30], e.g., the authors discuss numerical results obtained by using similar inequalities in the context of analyzing network reliability problems with 2-state links.

3.4 The Factoring Theorem

Our method for systematically generating the required s-disjoint pathsets and cutsets is based on extending the *factoring* theorem discussed, e.g., in [15, 30] in the context of evaluating the reliability of networks with 2-state (operate or fail) elements. Our extension concerns using the theorem with 3-state nodes, as required by the EXPO problem. We are not aware of other results that use the factoring theorem to analyze networks with k-state elements where $k \geq 3$.

To start, we introduce the following notation. Let G be a WSN where v is non-sink node, and $s_v \in \{rs, rns, fail\}$ be a possible state of v. Denote by $G \bullet (v, s_v)$ the constrained network G where node v is in state s_v. We now state the factoring theorem on 3-state nodes.

Theorem 3.2

$$\begin{aligned} Expo(G) = \; & p(v, rs) \cdot Expo(G \bullet (v, rs)) + \\ & p(v, rns) \cdot Expo(G \bullet (v, rns)) + \\ & p(v, fail) \cdot Expo(G \bullet (v, fail)) \end{aligned}$$

Proof. The theorem follows since the right hand side (RHS) exhausts all possible states of the node v. ■

Our method of using the theorem avoids enumerating all network states. Instead, we use the theorem to generate s-disjoint pathsets (in LB computations) and cutsets (in UB computations). The rationale is that the number of s-disjoint pathsets (or cutsets) is typically much smaller than the number of states in a network. Thus, the time spent in extending configurations to pathsets (or cutsets) pays off in terms of improving the running time of the algorithm. (A similar approach is used in [2] for computing lower bounds on a network reliability problem with 2-state nodes.) In addition, an effort is expended in generating pathsets (or cutsets) with minimal sets of nodes, so as to obtain good bounds.

We now introduce the following extended notation. First, we extend the notation $G \bullet (v, s_v)$ to the notation $G \bullet C$, where C is a configuration. The notation refers to a constrained network G where for each pair $(v, s_v) \in C$, node v is assigned state s_v.

Second, we assign integer values to the 3 states rs, rns, and $fail$ to use modulo 3 addition. For example, we set $rs = 0$, $rns = 1$, $fail = 2$. Thus, if s_i is a state and $k_i = 0, 1,$ or 2 is a shift amount then $s_i + k_i$ (mod 3) refers to a valid state.

Now, suppose that $C = ((v_1, s_1), (v_2, s_2), \cdots, (v_r, s_r))$ is a configuration on r nodes. Applying the factoring theorem to v_1 gives an expansion:

$$Expo(G) = \sum_{i=1}^{3} \Pr(C_i) \cdot Expo(G \bullet C_i)$$

where

$$\begin{aligned} C_1 &= ((v_1, s_1)) \\ C_2 &= ((v_1, s_1 + 1)) \\ C_3 &= ((v_1, s_1 + 2)). \end{aligned}$$

Subsequently, applying the theorem using v_2 to expand $Expo(G \bullet C_1)$ gives the expansion:

$$Expo(G) = \sum_{i=1}^{5} \Pr(C_i) \cdot Expo(G \bullet C_i)$$

where

$$\begin{aligned} C_1 &= ((v_1, s_1), (v_2, s_2)) \\ C_2 &= ((v_1, s_1), (v_2, s_2 + 1)) \\ C_3 &= ((v_1, s_1), (v_2, s_2 + 2)) \\ C_4 &= ((v_1, s_1 + 1)) \\ C_5 &= ((v_1, s_1 + 2)). \end{aligned}$$

Similarly, applying the theorem r times using all nodes of the configuration C gives the expansion:

$$Expo(G) = \sum_{i=1}^{2r+1} \Pr(C_i) \cdot Expo(G \bullet C_i)$$

where

$$\begin{aligned} C_1 &= ((v_1, s_1), \cdots, (v_{r-1}, s_{r-1}), (v_r, s_r)) \\ C_2 &= ((v_1, s_1), \cdots, (v_{r-1}, s_{r-1}), (v_r, s_r + 1)) \\ C_3 &= ((v_1, s_1), \cdots, (v_{r-1}, s_{r-1}), (v_r, s_r + 2)) \\ \cdots &= \cdots \\ C_{2r-2} &= ((v_1, s_1), (v_2, s_2 + 1)) \\ C_{2r-1} &= ((v_1, s_1), (v_2, s_2 + 2)) \\ C_2r &= ((v_1, s_1 + 1)) \\ C_{2r+1} &= ((v_1, s_1 + 2)) \end{aligned}$$

are pairwise s-disjoint configurations.

In LB computations, the algorithm chooses a pathset C generate configurations denoted C_1 to C_{2r+1} above. The algorithm then adds $\Pr(C_1)$ to the computed LB. Note that, $C_1 (= C)$ is a pathset. Subsequently, it chooses one of the configurations C_2 through C_{2r+1} to extend to a pathset, and use the new nodes of the computed pathset to generate other configurations. The computations terminate when no configuration admits a desired extension. In such case, the algorithm has exhausted a maximal set of s-disjoint pathsets, and the computed LB is the exact exposure value. A similar argument applies to UB computations by replacing pathsets with cutsets.

3.5 Main Algorithm

The overall algorithm is composed of 4 functions: function Main, function Factor, function extend_to_pathset, and extend_to_cutset explained in the next sections. We now explain the first two functions.

```
Algorithm 1: Function $\mathrm{Main}(G, k_{req})$:
Input: An instance of the EXPO problem
Output: The function generates pairwise s-disjoint configurations (either
        pathsets or cutsets) and returns the sum of their probabilities
Notation: $R$ is a table that stores the generated configurations.
1 Initialization: set
        $R[0] = (C = \emptyset, C_{new} = \emptyset, state = ACTIVE, p = 0.0)$
  (so, initially, $R$ has has only one record)
2 for $(iter = 1, 2, \ldots, maxIter)$ do
3     Let $i$ be the index of a configuration with largest probability in the heap
      $R.heap$; delete record $i$ from $R.heap$
4     Call function factor: $result = \mathrm{Factor}(R, i, k_{req})$
5     if$(result < 0)$ $R[i].state = BAD$
  end
6 return $\sum_{\substack{i = 0, 1, \ldots \\ \text{where } R[i].state \neq BAD}} R[i].p$
```

Function Main. This function takes as input an instance of the EXPO problem, and computes either a lower bound (LB), or an upper bound (UB) on the solution. If LB is specified, the function returns the sum of the probabilities of a set of s-disjoint pathsets. The pathsets are obtained using function extend_to_pathset (called from function Factor). If UB is specified, we use function extend_to_cutset and utilize

Algorithm 2: Function $\text{Factor}(R, i, k_{req})$

Input: Table R storing network configurations, and an index i of a configuration $R[i].C$ to use in factoring

Output: In record $R[i]$, if C can be extended to a pathset (or, cutset in UB computations) $C \cup C_{new}$ then apply factoring on the sequence of nodes in C_{new} to generate new s-disjoint configurations. For each generated configuration C', compute (if possible) an extension of C' to a pathset (or, cutset in UB computations). Store the record $(C', C'_{new}, ACTIVE, \Pr(C' \cup C'_{new}))$ in table R.

```
1  $result = \text{extend\_to\_pathset}(R[i].C, R[i].C_{new}, k_{req})$
2  if($result < 0$) return $result$
3  foreach (node-state pair $(v, s)$ in configuration $R[i].C_{new}$) do
4      for ($k = 1, 2$) do
5          Initialize ($C' = \emptyset, C'_{new} = \emptyset$)
6          Set $C' = (C, (v, s + k))$
7          $result = \text{extend\_to\_pathset}(C', C'_{new}, k_{req})$
8          if($result < 0$) continue
9          Set $p = \prod_{(v,s) \in (C' \bigcup C'_{new})} p(v, s)$
10         Insert record $(C', C'_{new}, ACTIVE, p)$ in table $R$
       end
11     Move tuple $(v, s)$ from $R[i].C_{new}$ to $R[i].C$
   end
12 Set $R[i].state = PROCESSED$
13 return $+1$
```

cutsets instead of pathsets. Table R is used to store the generated configurations. Each entry $R[i]$ stores a record of 4 fields:

– $R[i].C$: a generated network configuration

– $R[i].C_{new}$: if possible, an extension of C to a pathset (in LB computations), or cutset (in UB computations)

– $R[i].state$: a state of the record $R[i]$, $state \in \{ACTIVE, PROCESSED, BAD\}$. A record is $ACTIVE$ if it stores the empty initial configuration, or a non-empty configuration for which an augmentation to a pathset (or, cutset) has been computed and stored in C_{new}. A $PROCESSED$ record is an active record that has been used to generate other configurations using function Factor. A BAD record stores a configuration that can not be extended to a pathset (in LB computations), or cutset (in UB computations).

- $R[i].p$: for an $ACTIVE$ or $PROCESSED$ record, $p = \Pr(C \cup C_{new})$ (the probability of obtaining the stored pathset, or cutset). For a BAD record, $p = 0$.

The function also maintains a maximum heap storing indices in R. The top of the heap is an $ACTIVE$ record with highest probability p.

Step 1 initializes table R with an empty configuration $(C = \emptyset, C_{new} = \emptyset, state = ACTIVE, p = 0.0)$. Subsequently, Step 2 performs a prescribed number of iterations. In each iteration, Step 3 removes the top record from the heap, and Step 4 calls function Factor to process the removed record. Step 5 marks the record BAD if it can not be extended to a pathset (in LB computations), or cutset (in UB computations). Finally, Step 6 returns the sum of probabilities of all good configurations in R.

Function Factor. (Note: This function calls extend_to_pathset in Steps 1 and 7; these steps are changed to call function extend_to_cutset when UB computations are done.)

The function takes as input a record $R[i] = (C, C_{new}, ACTIVE, p)$ and computes in Step 1 an extension C_{new} so that $C \cup C_{new}$ is a pathset (in LB computations), or cutset (in UB computations). If $C \cup C_{new}$ satisfies this requirement, the loop in Step 3 performs factoring on each node-state pair in C_{new} to generate new s-disjoint configurations.

In more detail, in Step 3, if $R[i]$ has $C_{new} = ((v_1, s_1), \cdots, (v_r, s_r))$ then the loop in Step 3 generates $2r$ pairwise s-disjoint configurations. Processing the pair $(v_i, s_i) \in C_{new}$ is done during the ith iteration of Step 3 as follows. The iteration considers the prefix of $C \cup C_{new}$ composed of $(C, (v_1, s_1), \cdots, (v_{i-1}, s_{i-1}))$. Factoring on our target pair (v_i, s_i) creates two configurations:

$$(C, (v_1, s_1), \cdots, (v_{i-1}, s_{i-1}), (v_i, s_i + k)), \text{ for } k = 1, 2. \tag{3.1}$$

These two configurations are generated in the loop of Step 4.

Step 7 aims at extending each generated configuration C' to a pathset (or, cutset). If successful, Step 9 computes $p = \Pr(C' \cup C'_{new})$, and Step 10 inserts a new

ACTIVE record in table R. Step 11 prepares for a new iteration by moving the processed pair (v_i, s_i) from C_{new} to C.

Finally, after generating the $2r$ configurations, Step 12 marks record $R[i]$ as $PROCESSED$, and Step 13 returns with a success value $(+1)$.

The algorithm can be shown to be correct by showing that function Main generates pairwise s-disjoint configurations in table R.

3.5.1 Correctness

We now show that

Theorem 3.3 *Function* Main *generates pairwise s-disjoint configurations in table* R*.*

Proof. We induct on the number of iterations done by the function. The basis when R has one entry is straightforward. For any given iteration $m \geq 1$, assume the theorem holds at the start of the iteration. We show that it holds true after the iteration. Let $R[i]$ be the record selected for expansion in the iteration, and let $R[j]$ be any other record in table R. By the induction hypothesis, $R[i].C$ and $R[j].C$ differ in the state of at least one node that occurs in both configurations.

Function Factor generates new configurations by factoring on nodes in $R[i].C_{new}$. Each of the new configurations has the prefix $R[i].C$. Thus, each newly generated configuration is s-disjoint from any configuration stored in R. In addition, any two configurations generated by Factor differ in the state of at least one node in $R[i].C_{new}$. So, any pair of the newly generated configurations is s-disjoint. Thus, all configurations stored in the table R after iteration m are pairwise s-disjoint, as required. ■

3.6 Pathset Augmentation

In this section we illustrate by an example the main ideas behind the design of function $\mathrm{extend\_to\_pathset}$. The function takes as input a configuration C on a subset $V(C) \subseteq V$ of nodes, and aims at computing an extension configuration

C_{new} of C such that **(a)** nodes of C and C_{new} are disjoint, **(b)** $C \cup C_{new}$ is a pathset of G, and **(c)** $\Pr(C \cup C_{new})$ is as high as possible.

The function strives to achieve condition **(c)** by two means: first, it computes a minimal extension C_{new} that satisfies conditions **(a)** and **(b)**, and second, it tries to optimize the selection of nodes added to C_{new}. Roughly speaking, the main steps are as follows.

1. Given an input configuration C, we mark all nodes in $V \setminus V(C)$ as $free$ nodes (these nodes are not assigned any specific state thus far).

2. The algorithm then executes iteratively. In each iteration it tries to identify the following:

   (a) A sensing capable node that does not reach the sink in the currently computed configuration $C \bigcup C_{new}$, denoted $best$. Note that node $best$ can be a rs node in configuration C that does not reach the sink in $C \bigcup C_{new}$.

   (b) A path, denoted $P_{best\_to\_sinktree}$, that connects node $best$ to some node that can reach the sink in the currently computed configuration $C \bigcup C_{new}$

   (c) Node $best$ can be assigned the state rs, and each free node on the path $P_{best\_to\_sinktree}$ can be assigned either state rs or rns such that $\Pr(P_{best\_to\_sinktree})$ is maximum over all possible choices of node $best$.

3. Thus, adding all nodes in the configuration specified by $P_{best\_to\_sinktree}$ increases the number of nodes of type rs connected to the sink. The function terminates successfully if the number of nodes in state rs in $C \bigcup C_{new} \geq k_{req}$. Else, the function returns with failure (-1).

Example. Fig. 3.2 illustrates an instance of the EXPO problem on $|V| = 8$ nodes where $k_{req} = 3$. The figure shows layers 1, 2, and 3 of a breadth first search (BFS) tree rooted at the sink. The figure also illustrates an input configuration

$$C = ((x_1, rns), (x_2, rns), (x_3, rns), (y_1, rs), (z_2, rs)).$$

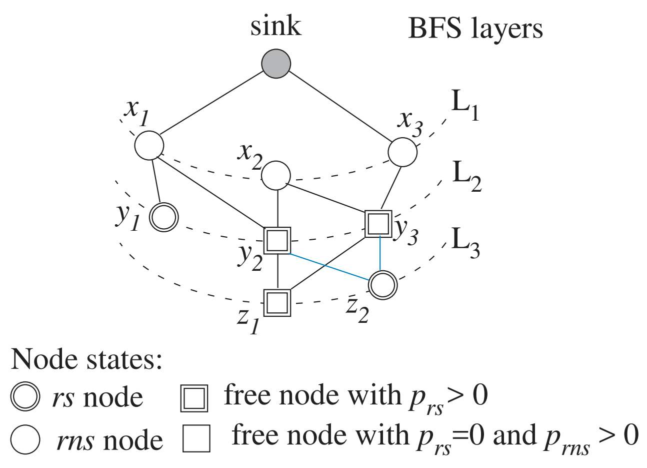


Figure 3.2: An example of pathset augmentation for the EXPO problem

The remaining 3 nodes y_2, y_3, and z_1 are free nodes.

Note that C is not a pathset since only node y_1 is in state rs and can reach the sink while the other rs node in C (node z_2) can not reach the sink in the configuration. Now, suppose that

- Node y_2 has $p_{relay}(y_2) = 0.2$ and $p_{sense}(y_2) = 0.1$ (i.e., $p_{rs}(y_2) = .02$ and $p_{rns}(y_2) = .18$). Thus, to extend C by adding (y_2, rs), the best path is $P_{y_2\_to\_sinktree} = (y_2)$ with probability .02.

- Node y_3 has $p_{relay}(y_3) = 0.9$ and $p_{sense}(y_3) = 0.1$ (i.e., $p_{rs}(y_3) = .09$ and $p_{rns}(y_3) = .81$). Thus, to extend C by adding (y_3, rs), the best path is $P_{y_3\_to\_sinktree} = (y_3)$ with probability .09.

- Node z_1 has $p_{relay}(z_1) = 0.7$ and $p_{sense}(z_1) = 0.8$ (i.e., $p_{rs}(z_1) = .56$ and $p_{rns}(z_1) = .14$). Thus, to extend C by adding (z_1, rs) the best path is $P_{z_1\_to\_sinktree} = (z_1, y_3)$ with probability $= 0.56 \times 0.81 \approx 0.45$.

The algorithm selects node $best = z_1$, sets the path $P_{best\_to\_sinktree} = (z_1, y_3)$, and adds to C_{new} the pairs $((z_1, rs), (y_3, rns))$. Note also that in the new configuration $C \cup C_{new}$, node z_2 is in state rs and reaches the sink. We conclude that since $k_{req} = 3$, $C \cup C_{new}$ is a pathset. ■

We note that in step 2.c if each free node on the identified path $P_{best\_to\_sinktree}$ with $p_{sense} > 0$ is assigned the rs state, the function will always succeed in finding an extension C_{new} if a solution exists.

3.7 Cutset Augmentation

In this section we illustrate by an example the main ideas behind the design of function $\mathrm{extend\_to\_cutset}$. Analogous to function $\mathrm{extend\_to\_pathset}$, function $\mathrm{extend\_to\_cutset}$ takes as input a configuration C on a subset $V(C) \subseteq V$ of nodes, and aims at computing an extension configuration C_{new} of C such that **(a)** nodes of C and C_{new} are disjoint, **(b)** $C \bigcup C_{new}$ is a cutset of G (recall that a cutset is a configuration that can not be extended to a pathset), and **(c)** $\Pr(C \bigcup C_{new})$ is as high as possible.

Function $\mathrm{extend\_to\_cutset}$ strives to achieve condition **(c)** by computing a minimal extension C_{new} that satisfies conditions **(a)** and **(b)**. The function relies on the following methods.

1) A method to test whether any given input configuration C_{in} is a cutset

2) A method to select a free node not in C_{in} to flip its state to failed (if possible else convert the state to rns) and add it to C_{new}

Methods 1 and 2 are applied iteratively until $C \cup C_{new}$ forms a cutset (if such cutset exists).

3) A method that aims at minimizing the size of the computed C_{new}

Method 1 works by assuming that each non-sink node x in $V \setminus C_{in}$ is in state rs (if $p_{rs}(x) > 0$), or state rns (if $p_{rns}(x) > 0$, but $p_{rs}(x) = 0$), and testing whether the network is operational. The method concludes that C_{in} is a cutset iff this assumed network state is failed.

Work of Method 2 is guided by inputting a total order $(v_1, v_2, \cdots, v_n)$ of V. The method scans this sequence, selects the first free node, converts its state to failed (if possible else convert the state to rns), and appends the node to C_{new}. We experimented with a total order derived from a BFS layering of G rooted at the sink node (as shown in the next example).

Method 3 is motivated by the observation that C_{new} constructed by repeated applications of Methods 1 and 2 may not be minimal. The method works by testing whether the state of each node in the constructed sequence C_{new} can be reverted from failed or rns to free without losing the cutset property. If so, the state of any such node is reverted, and the node is removed from C_{new} The order of testing the nodes is the same order of adding nodes to C_{new}.

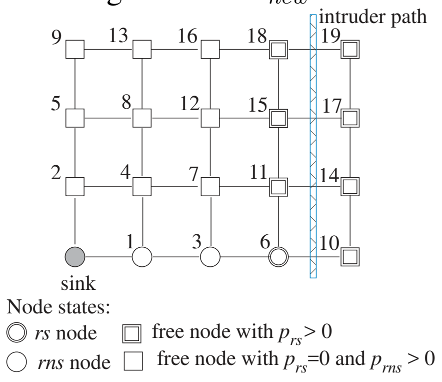


Figure 3.3: An example of cutset augmentation for the EXPO problem

Example. Fig. 3.3 illustrates an instance of the EXPO problem where $k_{req} = 3$. The figure also illustrates an input configuration on 3 nodes: $C = ((1, rns), (3, rns), (6, rs))$ to be extended to a cutset. We assume that the nodes are ordered as $(1, 2, \cdots, 19)$ according to a BFS layering rooted at the sink. Applying Methods 1 and 2 iteratively using the above total order results in computing C_{new} with the node sequence $(2, 4, 5, 7, 8, 9, 10, 11)$. Applying Method 3 to the above sequence results in only failing the nodes of the sequence $(7, 8, 9, 10, 11)$. Thus, the function returns $C_{new} = ((7, fail), (8, fail), (9, fail), (10, fail), (11, fail))$. ■

3.8 Numerical Results

In this section, we present simulation results to illustrate the potential benefits of using our devised algorithm. We present results on $W \times L$ grid networks that are representative of what we obtained with other topologies. A $W \times L$ grid network has W rows at coordinates $y = 0, 1, \cdots, W - 1$, and L columns at coordinates $x = 0, 1, \cdots, L - 1$. In some experiments the sink is located at the origin, and the intruder path P runs vertically midway between the last two columns. For such path P, a node x that lies immediately on either side of P has non-zero $p_{sense}(x)$ (else, $p_{sense}(x) = 0$).

We recall that our algorithm computes the exposure function for any given distribution of the p_{sense} probabilities. Our choice of a line segment path P simplifies the setting of the p_{sense} probabilities.

3.8.1 Exact Exposure Computations

Using the nodes of a pathset to derive the factoring process allows the program to compute $Expo(G, k_{req})$ exactly without exhausting all possible 3^n configurations of the network. The savings make the exact computations practical for small networks (e.g., $W{\times}L$ grids where $W, L \leq 6$), as illustrated in Table 3.1. All computations are done on a personal computer and finish in 15 seconds, or less.

Table 3.1: Exact exposure computations

Grid dimensions $W{\times}L$	$k_{required}$	**Maximum number of configurations**	**configurations examined by algorithm**
2×3	1	3^5	12
3×3	2	3^8	148
3×4	2	3^{11}	378
4×4	2	3^{15}	2120

3.8.2 Gaps Between Lower and Upper Bounds

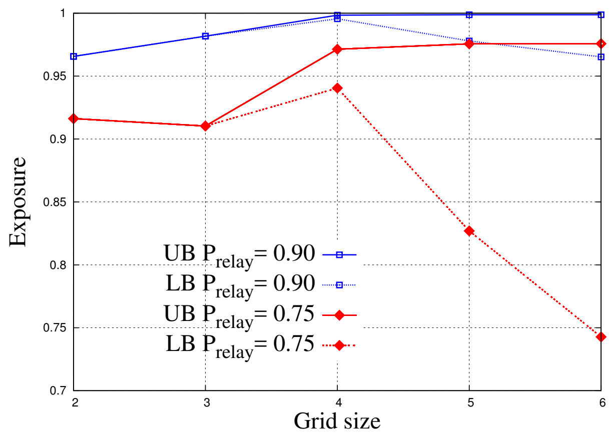


Figure 3.4: Exposure versus size (varying p_{relay})

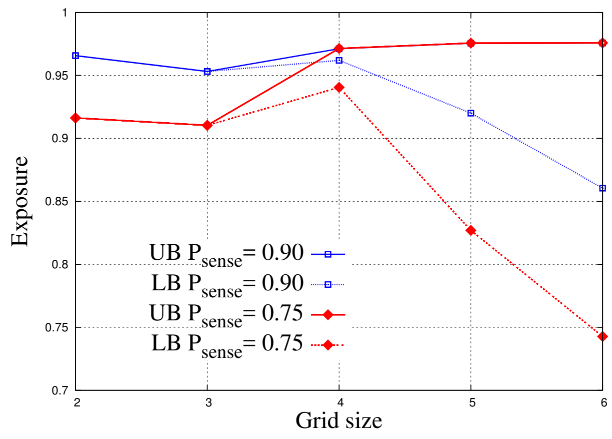


Figure 3.5: Exposure versus size (varying p_{sense})

Here, we present results on the gap between the obtained lower and upper bounds. We use square $W \times W$ diagonal grids where $W \in [2, 6]$, the sink is located at the origin, and the intruder path runs vertically between the last two columns. In Fig. 3.4, we report both lower bounds (LBs) and upper bounds (UBs) when the algorithm is repeated for 1000 iterations, $p_{sense} = 0.75$ for nodes exposed to the intruder path (else, $p_{sense} = 0$), and $p_{relay} \in \{0.75, 0.90\}$. In Fig. 3.5, we report lower and upper bounds when the algorithm is repeated for 1000 iterations, $p_{relay} = 0.75$ for all nodes, and for nodes exposed to the intruder path $p_{sense} \in \{0.75, 0.90\}$ (else, $p_{sense} = 0$).

We remark that for each choice of (p_{relay}, p_{sense}), the gap increases with the size of the network. This is expected since larger networks have more nodes, and the path P is farther away from the sink node. So, larger networks have more relevant configurations that are not taken into consideration when using 1000 iterations.

Also, for the (p_{relay}, p_{sense}) settings shown in the figures, the obtained LBs on the exposure function achieved by the entire network is comparable to the p_{relay} and p_{sense} of individual components. For example, in Fig. 3.4, the computed LB on a 6×6 grid when ($p_{relay} = 0.75$, $p_{sense} = 0.75$ for exposed nodes) is about 0.75. This shows the possibility of building a large network whose performance is at least as good as the performance of its individual components.

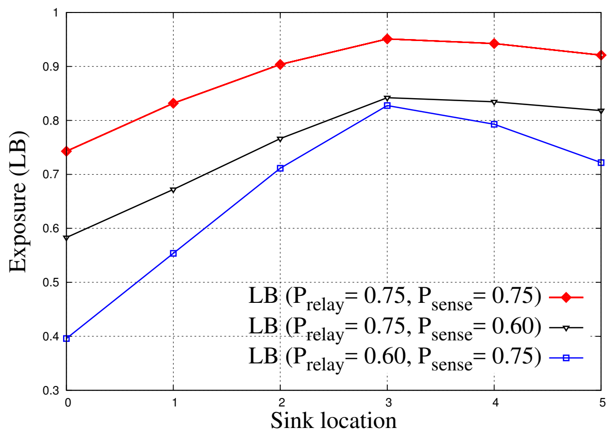


Figure 3.6: Exposure versus sink location

3.8.3 Optimizing Sink Location

Given a WSN and a set $\{P_1, P_2, \cdots, P_k\}$ of possible intruder paths, a designer may wish to position the sink node so as to optimize an objective function on the set $\{Expo(G, P_i) : i = 1, 2, \cdots, k\}$ (say, e.g., maximize the minimum exposure of all paths). Our algorithm enables the derivation of lower bounds on each of the required k exposure functions, and hence can be used to derive lower bounds on the devised objective function. Hence, our algorithm provides a way to search for the best position of the sink node.

To gain more insight into a concrete situation, we use a 6×6 doubly diagonal grid to measure the exposure of an intruder's path passing vertically between the last two columns when $k_{req} = 3$. Here, we experiment with locating the sink node at different nodes on the diagonal (nodes with coordinates (i, i) for $i = 0, 1, \cdots 5$).

Fig. 3.6 illustrates the obtained results. We note that the exposure function increases monotonically in the interval $[0, 3]$. However, past point $(3, 3)$, the function starts to decrease even though the shortest distance between the sink and the closest point on P occurs when the sink is at locations $(4, 4)$ and $(5, 5)$. This behaviour can be explained by a careful analysis of the number of nodes that can sense the intruder path and lie at a certain distance from the sink node. To simplify the comparison, we write such numbers as a vector $(n_1, n_2, n_3, \cdots)$ to indicate that there are n_i such nodes at distance i from the sink. When the sink is located at $(3, 3)$ there are 12 nodes that can sense the intruder path. The 12 nodes are distributed as $(3, 7, 2)$. On the other hand, when the sink is located at $(5, 5)$ there are 11 nodes that can sense the intruder (we assume the sink does not participate in sensing). The 11 nodes are distributed as $(3, 2, 2, 2, 2)$. These vectors show the clear advantage of putting the sink at location $(3, 3)$.

3.8.4 Intentional Jamming

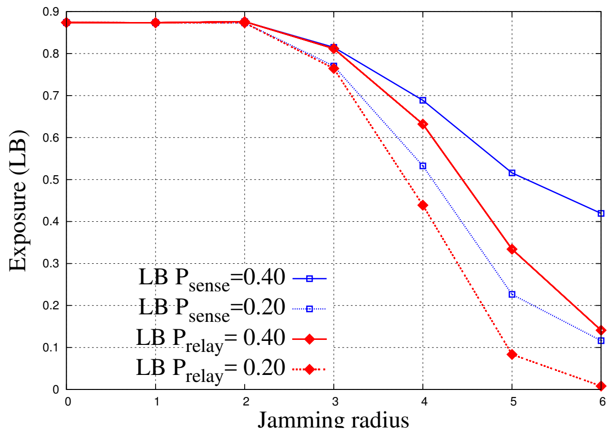


Figure 3.7: Exposure versus jamming radius

Intruders may use devices that can partially jam the communication and/or the sensing abilities of nodes surrounding the device. A network designer may be interested in estimating the detrimental effect of the intruder positioning such devices at certain places along his path across the network (e.g., throwing the device upon entrance in the sensor field, or mid-way on the path, and so on).

To analyze such effects, we present results when a jamming device is positioned at coordinates $(5, 5)$ in 6×6 grid with sink at origin, and $k_{req} = 3$. The radius of effect, R_{jam} of the device is varied in the range $R_{jam} \in [0, 6]$. When $R_{jam} = 0$, the device affects only the node at its current position (i.e., node $(5, 5)$). When $R_{jam} = 1$ the device affects 3 nodes at coordinates $(4, 5)$, $(5, 4)$, and $(5, 5)$, and so on.

In Fig. 3.7, two curves illustrate the degradation in $Expo(G, P)$ when p_{sense} of the affected nodes is reduced from 0.8 to either 0.4 or 0.2. The two other curves show the degradation in $Expo(G, P)$ when the p_{relay} of the affected nodes is reduced from 0.8 to either 0.4 or 0.2. In general, the results show that jamming communication has more impact than jamming sensing. Additionally, the results show network robustness when $R_{jam} \leq 2$. This result is worth noting since many nodes are affected when $R_{jam} = 2$.

3.9 Concluding Remarks

In this chapter, we consider WSNs where nodes can succeed or fail in sensing and reporting intrusion events with known probabilities. We introduce the EXPO problem that calls for evaluating the likelihood that a prescribed number k_{req} of nodes succeed in sensing and reporting intrusion to a sink node. Our devised algorithm for solving the problem is based on extending a factoring theorem to networks with 3-state nodes, complemented with methods for computing most probable pathsets and cutsets. The algorithm computes lower and upper bounds on a solution in any given number of iterations.

The obtained simulation results show that our algorithm provides practical means of computing exact solutions on small networks. For large networks, the obtained results show that the algorithm provides useful means of analyzing the effect of making topology changes to the network such as changing sink location, and assessing the deterioration in performance when jamming devices are used.

Chapter 4

Breach Path Reliability for WSNs

In the previous chapter, we consider an intrusion detection reliability problem when the intruder path P is specified in the given problem instance, and $k_{req} \geq 1$. In this chapter, we consider another intrusion detection reliability problem where the intruder can follow any path from a specified set of paths, and $k_{req} = 1$. We refer to this latter problem as the breach path detection reliability (BPDREL) problem.

The BPDREL problem formulation assumes that the area under surveillance is 2-dimensional and bounded by a polygon with known sides. The BPDREL problem calls for computing the network's success probability in detecting an intruder that crosses the perimeter through any specified subset of the available entry-exit polygon sides. We adopt the unified framework for computing lower and upper bounds to the BPDREL problem. This is done by designing suitable algorithms for the formalized $E2P$ and $E2C$ problems. The algorithm can handle WSNs with arbitrary sets of links where sink nodes are placed in arbitrary positions. Convergence to an exact solution is guaranteed as the number of performed iterations is increased.

Some of the results in this chapter appear in [19].

4.1 Problem Formulation

In this section, we introduce some basic definitions needed to formulate the BPDREL problem and state some of its properties.

4.1.1 Network and Intrusion Models

The WSN is modelled by an undirected graph $G = (V \cup \{s\}, E)$ where s is a sink node. Communication links in E are bidirectional. A link $(x, y) \in E$ is considered in the network only if it satisfies the following condition: node x and/or node y can sense any object crossing the line segment (x, y). Failing to satisfy this condition makes a link unable to guarantee detection of objects crossing the corresponding line segment, and hence the link is not used in solving the problem.

WSNs dealt with in the BPDREL problem can be embedded in the plane so that the network's perimeter is defined by a polygon with well defined sides. We denote the nodes on the perimeter by X. Each intruder path (also called a breach path) passing across the network is associated with a pair (d^{in}, d^{out}) of entry-exit sides of the perimeter. Any such path is bidirectional with interchangeable entry and exit sides. Without loss of generality, it is assumed that any such path touches only the entry and exit sides of the perimeter, and no other side. The BPDREL problem deals with any given set of entry-exit pairs of sides $D = \{(d_i^{in}, d_i^{out}) : i = 1, 2, 3, \ldots\}$, $d_i^{in} \neq d_i^{out}$, that we want to guard against using them by an intruder. The term **D-attack** refers to an intrusion event that uses any pair of sides in D.

4.1.2 Node Failure

Nodes in the problem are prone to failure due to operation in harsh environments. For a node v, $p(v)$ denotes the node's operating probability during some interval of time. We use $q(v) = 1 - p(v)$ to denote the node's failure probability. The sink s is assumed to be perfectly reliable with $p(s) = 1$. We assume nodes fail independent of each other. A random failure event leaves a network in some state S where some nodes are operating and the remaining nodes are failed. We use S to refer to both a network state and the set of operating nodes in that state. Thus, a network on n nodes has 2^n states. In addition, a state S occurs with probability $\Pr(S) = \prod_{v \in S} p(v) \prod_{v \notin S} q(v)$.

In the BPDREL problem, a network state S is *operating* if and only if it guarantees the detection of any D-attack. Equivalently, S is operating if and only if any possible intruder path crossing the polygonal perimeter using any pair of entry-exit sides in D is intersected by a fully operating path from some node to the sink. The problem defines the network's reliability as

$$Rel(G) = \sum \Pr(S : S \text{ is an operating state with respect to } D) \tag{4.1}$$

Definition (the BPDREL problem). Let $G = (V \cup \{s\}, E)$ be a WSN where each node v, $v \neq s$, has an operating probability $p(v)$. Each point on a line segment $(x, y) \in E$ can be detected by either x and/or y. In addition, let $D = \{(d_i^{in}, d_i^{out}) : i = 1, 2, 3, \ldots\}$, $d_i^{in} \neq d_i^{out}$, be a set of entry-exit pairs of sides on the perimeter of G. Find the probability $Rel(G, p, D)$ that G is in a state that ensures that any D-attack is detected. ■

The BPDREL problem has been investigated in [45, 46]. In [46], the authors develop an exact algorithm on the class of grid networks, and the sink node is located on the network corner. In [45], the authors generalize their results in [46] by introducing an algorithm for WSNs that can be embeded in a grid-like network with a corner sink node. In contrast, our work in this chapter concerns deriving lower and upper bounds for arbitrary WSNs with unconstrained sink locations.

4.1.3 Complexity Analysis

Theorem 4.1 *The* BPDREL *problem is #P-hard.*

Proof. We recall from theorem 3.1, that in [1] the authors have shown that computing the wireless $2REL$ problem is #P-Complete even when the problem is restricted to a probabilistic partial grid networks $G = (V, E)$ with equal node operating probabilities and equal communication ranges. An instance (G, s, t) of the $2REL$ problem can select s and t as any two non-adjacent nodes in G. The reduction in [1] allows us to assume that s and t are two non-adjacent perimeter nodes on the partial grid network G.

To show that BPDREL is #P-hard, we reduce in polynomial time a given instance (G, s, t) of the $2REL$ problem on wireless partial grid networks to an instance (G, p, D) of the BPDREL problem such that $Rel(G, s, t) = Rel(G, p, D)$, as follows:

1. The probabilistic graph G (with its associated node operating probabilities) of $2REL$ instance is the same for the constructed BPDREL instance.
2. Node t is taken as the sink node of the BPDREL instance.
3. In BPDREL, $D = \{d^{in}, d^{out}\}$ where d^{in} and d^{out} are the two sides of the perimeter of the partial grid network G incident to node s.

One can then verify that $Rel(G, s, t) = Rel(G, p, D)$, as claimed. ■

4.2 Protection Intervals, Pathsets, and Cutsets

In this section we extend the concept of *protection intervals* introduced in [46] to deal with our current problem where the sink can be placed in any arbitrary location in the network. We also introduce the concepts of a network *configuration*, *pathset*, and *cutset* needed to present our bounding methods. Throughout the presentation, we characterize some of these concepts in terms of protection intervals.

4.2.1 Protection Intervals

Let (G, p, D) be an instance of the BPDREL problem. Denote by $X \subseteq V$ the nodes on the WSN's perimeter, and let $(d^{in}, d^{out}) \in D$. We need the following definition.

Definition ((d^{in}, d^{out})-protection interval(s)). The protection interval(s) of (d^{in}, d^{out}) is defined as follows. Traverse the perimeter in a clockwise direction, and write X as a sequence:

$$(first(d^{in}), second(d^{in}), x_i, x_{i+1}, \dots, \\ first(d^{out}), second(d^{out}), x_j, x_{j+1}, \dots)$$

where $first(d)$ and $second(d)$ denote the two end points of a side d. Set the intervals X_1 and X_2 to the sequences:

$$X_1 = (second(d^{in}), x_i, x_{i+1}, \ldots, first(d^{out})), \text{ and}$$
$$X_2 = (second(d^{out}), x_j, x_{j+1}, \ldots, first(d^{in})).$$

We say that X_1 (or X_2) is a protection interval of (d^{in}, d^{out}) if it does not contain the sink s. ■

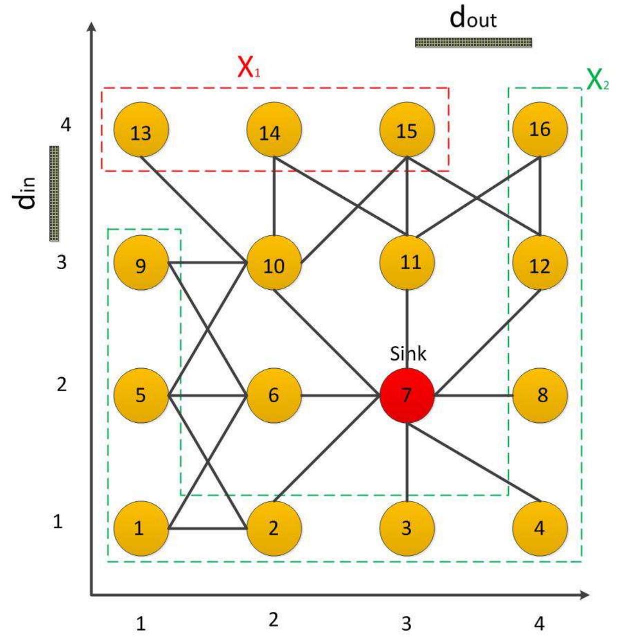


Figure 4.1: An instance of the BPDREL problem

Example. In Fig. 4.1, consider the entry-exit sides $(d^{in} = (9, 13), d^{out} = (15, 16))$. The sink is at (x, y)-coordinates of $(3, 2)$, and the corresponding (d^{in}, d^{out})-protection intervals are $X_1 = (13, 14, 15)$, and $X_2 = (16, 12, \ldots, 5, 9)$. Moving the sink to coordinates $(1, 1)$ results in having only one protection interval X_1 (since X_2 contains the sink and hence, by definition, X_2 is not a protection interval). ■

Given a network state S, we say that a protection interval X_i is *covered* in S if there exists a path of operating nodes from some node on X_i to the sink. Our interest in covering (d^{in}, d^{out})-protection interval(s) stems from the following remark: a state S of the network guarantees that any (d^{in}, d^{out})-intruder path is detected if and only if S covers the associated protection interval(s). This remark follows since the operating path(s) that covers the intervals intersects all possible (d^{in}, d^{out})-intruder paths.

As mentioned in [45], this observation can be extended to the general case where $D = \{(d_i^{in}, d_i^{out}) : i = 1, 2, 3, \ldots\}$, and $\mathbf{X}$ is the corresponding set of protection intervals, as summarized below.

Lemma 4.1 *Given an instance (G, p, D) of the* BPDREL *problem. A network state S of G is operating (i.e., S guarantees the detection of any D-attack) if and only if each protection interval $X_i \in \mathbf{X}$ is covered.* ■

In [46], the following remark has also been mentioned: if $\mathbf{X}$ has two intervals X_i and X_j where $X_i \supset X_j$ then X_i can be deleted from $\mathbf{X}$ without violating the above lemma. Thus, it suffices to consider a *reduced* set of intervals obtained by iteratively deleting an interval X_i form $\mathbf{X}$ if it contains another interval X_j. We henceforth use the $(G, p, \mathbf{X})$ to refer to a given instance of the BPDREL problem where $\mathbf{X}$ is a reduced set of protection intervals.

4.2.2 Pathsets and Cutsets

In this part, we define the concepts of pathsets and cutsets for the BPDREL problem.

Pathsets. Given an instance $(G, p, \mathbf{X})$ of the BPDREL problem, we define a *pathset* to be any operating configuration C. By Lemma 4.1, C is pathset if and only if all protection intervals $\mathbf{X}$ are covered in C.

Example. In Fig. 4.1, configuration $C = ((12, op), (15, op))$ is operating with respect to the entry-exit pair of sides (d^{in}, d^{out}) regardless of the state of the remaining nodes. Note that node 12 is connected to the sink node 7. Hence, C is a pathset. ■

We also find it useful to define the following notion of parameterized pathsets. For any subset of protection intervals $\mathbf{X}' \subseteq \mathbf{X}$, define an $\mathbf{X}'$-pathset to be a configuration C that covers each interval in $\mathbf{X}'$. Thus, pathsets are equivalent to $\mathbf{X}$-pathsets.

Cutsets. Given an instance $(G, p, \mathbf{X})$ of the BPDREL problem, a *cutset* is a configuration C that can not be extended to a pathset. The definition of a cutset is parameterized as follows. For any subset of protection intervals $\mathbf{X}' \subseteq \mathbf{X}$, define an $\mathbf{X}'$-cutset to be a configuration C in which the failed nodes forbid the remaining nodes to cover all intervals in $\mathbf{X}'$. When $\mathbf{X}'$ has one interval X_i, we abbreviate $\{X_i\}$-cutset to X_i-cutset. For example in Fig. 4.1, $C = ((10, fail), (11, fail), (12, fail))$ is a cutset that isolates nodes in protection interval X_1 from reaching the sink node 7.

Bounds from Pathsets and Cutsets. Our method of obtaining a lower bound on $Rel(G)$ for the BPDREL problem is to compute a set of pairwise s-disjoint pathsets $\{P_1, P_2, \cdots, P_r\}$, and observe that $Rel(G) \geq \sum_{i=1}^{r} \Pr(P_i)$ where the summation on the RHS constitutes the intended lower bound.

Similarly, to obtain an upper bound, we compute a set of pairwise s-disjoint cutsets $\{U_1, U_2, \cdots, U_r\}$, and observe that $Rel(G) \leq 1 - \sum_{i=1}^{r} \Pr(U_i)$ where the RHS is the desired upper bound. The main algorithm (Section 4.5) presents a method for generating pairwise s-disjoint pathsets and cutsets. The method relies on generating configurations and extending a selected subset of these configurations to pathsets (or cutsets). The extension process is critical to the quality of the obtained bounds.

4.3 Extension to a Pathset

In this section, we define the optimal *extension to a pathset* (E2P) problem as follows. Given an instance $(G, p, \mathbf{X})$ of the BPDREL problem, and a configuration C that is not a cutset, extend C to a pathset $C \bigcup C_{new}$ so that $\Pr(C \bigcup C_{new})$ is as high as possible. Nodes in C and C_{new} form disjoint sets.

Our contribution in this section is to show that for networks with arbitrary edge sets, the E2P problem is NP-complete even if all operating probabilities are equal. This hardness result motivates the search for useful effective heuristic solutions. We next present one such solution that exploits some properties of the problem.

4.3.1 Complexity of the E2P Problem

We now show that

Theorem 4.2 *The decision version of the E2P problem on graphs with arbitrary edge sets is NP-complete even if all nodes have equal probabilities.*

Our proof utilizes the *Exact Cover by 3-Sets* (X3C) problem which is a well known NP-complete problem (see, e.g., problem [SP2] in [28]). An instance of the X3C problem is a pair (X, Y) where X is a set of $3q$ elements, $q > 0$, and Y is a set of 3-element subsets of X. The problem is to decide whether Y contains a subset Y', $|Y'| = q$, that covers all elements in X.

Proof. We transform in polynomial time any given instance (X, Y) of the X3C problem to an instance $(G, p, \mathbf{X}, C)$ of the E2P problem such that the X3C problem has a solution if and only if the E2P problem has a solution of some specified probability. In particular, we construct the E2P instance as follows.

- The graph G has a set $V(G) = \{\{s\} \bigcup Y \bigcup X\}$ of nodes where s is a new node (the sink node of G), and nodes in X and Y are in one-to-one correspondence with the members of sets X and Y. In G, node s is made adjacent to each node in Y, and node $x_i \in X$ is adjacent to a node $y_j \in Y$ iff element x_i appears in set y_j.

- G is embedded in the plane to form a network with nodes $(s, x_1, x_2, \cdots, x_{3q})$ forming a polygonal perimeter with $3q + 1$ sides. Note that the sides of the polygon does not necessarily correspond to communication or sensing links in G. Rather, the sides are used only to define the entry-exit sides the intruder may use.

- We assign a non-zero operating probability p to each node in $X \bigcup Y$.

- We define $3q + 1$ entry-exit pairs of sides in the constructed BPDREL problem so that each node x_i forms a protection interval. Thus, $\mathbf{X} = X$ is a set of $3q$ pairwise disjoint protection intervals.

- We set the input configuration $C = \emptyset$ in the E2P problem.

Thus, by definition, a pathset of the constructed E2P instance is a set of nodes that connects each node $x_i \in X$ to the sink s. Since $|X| = 3q$, and each node in Y is connected to exactly 3 nodes in X, a highest probability pathset configuration C_{new} has $\Pr(C_{new}) = p^{4q}$.

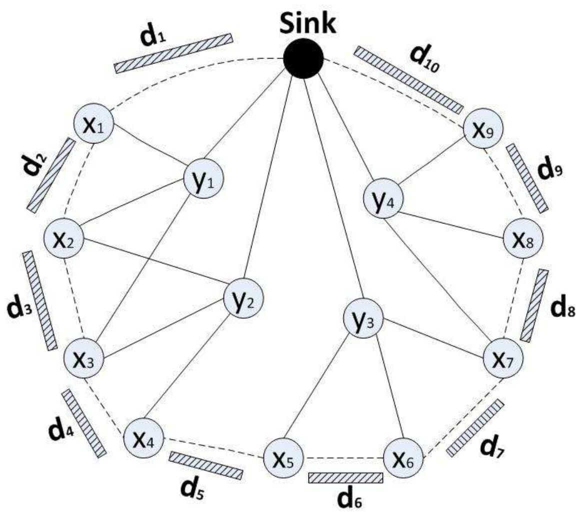


Figure 4.2: Reduction to the E2P problem

It is then easy to see that the X3C has a solution if and only if the E2P has a solution of probability at least p^{4q}. Lastly, membership in NP can be verified since in polynomial time one can:

a) non-deterministically guess a set of nodes C_{new}, and

b) deterministically verify that $C \bigcup C_{new}$ is a pathset, and $\Pr(C_{new}) \geq P_{threshold}$ where $P_{threshold}$ is given in the decision version of the E2P problem.

■

We remark that the graph G constructed in the proof contains a set of edges that may not arise in a typical wireless network. Our proof, however, shows that the running time of any exact algorithm for solving the E2P problem is likely to be exponential unless it utilizes properties intrinsic to graphs arising from wireless networks.

4.3.2 An E2P Heuristic Algorithm

The complexity of the E2P problem motivates the search for an efficient heuristic algorithm that exploits some properties of the problem. Function E2P highlights the structure of our proposed algorithm. To explain the algorithm, we introduce the following definitions. The definitions refer to the input configuration C, and any

Algorithm 3: Function E2P$(G, p, \mathbf{X}, C, C_{new})$

```
Input: An instance $(G, p, X, C)$ of the $E2P$ problem
Output: If the $E2P$ has a solution return $+1$, and a solution in $C \cup C_{new}$. Else
        (if C is a cutset), return $-1$.
1  if($C$ is a cutset) return $-1$
2  Set $C_{new} = \phi$
3  Set $\mathbf{X}_{unc} = \mathbf{X}$ minus intervals covered by operating nodes that can reach the
     sink in $(C \cup C_{new})$.
4  while ($\mathbf{X}_{unc} \neq \emptyset$) do
5      Set $G' = logp\_distance(G, C \cup C_{new})$
6      Compute shortest paths in $G'$ from sink $s$ to each node in $V(\mathbf{X}_{unc})$.
         Denote the shortest distance to $v_i$ by $cost(v_i)$.
7      Choose among the nodes in $V(\mathbf{X}_{unc})$ a node $v_{best}$ with smallest
         cost-effectiveness value $\alpha(v_{best})$.
8      Set $P_{best}$ to the shortest path from $v_{best}$ to $s$ in $G'$
9      Add nodes in $P_{best}$ that are not in $C \bigcup C_{new}$ to $C_{new}$ as operating nodes
10     Update $\mathbf{X}_{unc}$ as in Step 3
   end
11 return $+1$
```

valid extension configuration C_{new}.

Definitions.

- $connected\_to(s, C \bigcup C_{new})$: a set of nodes composed of the sink s and all other operating nodes in $C \bigcup C_{new}$ that can reach s by a path of operating nodes in $C \bigcup C_{new}$

- $\mathbf{X}_{unc}$: a subset of the $\mathbf{X}$ intervals that are not covered by any node in $connected\_to$ $(s, C \bigcup C_{new})$

- $covered\_by(v, \mathbf{X}_{unc})$: the subset of intervals of $\mathbf{X}_{unc}$ covered by a perimeter node v connected to the sink by a path of operating nodes.

- $logp\_distance(G, C \bigcup C_{new})$: a weighted graph obtained from G as follows: if a node v is assigned the failed state in configuration $C \bigcup C_{new}$ then delete v from G. Else, if v is assigned an operating state in $C \bigcup C_{new}$ then set the weight $w(v) = 0$, else set $w(v) = -\log p(v)$. Thus, all weights of G are positive real numbers. ■

Step 1 returns with failure (-1) if C is a cutset. Step 2 initializes $C_{new} = \emptyset$. Step 3 identifies the subset $\mathbf{X}_{unc} \subseteq \mathbf{X}$ of intervals that remain to be covered. The loop in Step 4 iterates until C_{new} is augmented with enough operating nodes to cover all intervals in $\mathbf{X}_{unc}$. The body of the loop makes use of the following observations.

First, we observe that for any non-failed node v in G, the most reliable paths between v and some node in $connected\_to(s, C \bigcup C_{new})$ corresponds to the shortest (s, v)-paths in G' (where node weights in G' are viewed as distances). This follows since if $P_{s,v}$ is any (s, v)-path then $-\log \Pr(P_{s,v}) = \sum_{v \in V(P_{s,v})} w(v)$. In Step 6, we use a single-source shortest paths algorithm (see, e.g., [16]) to compute the shortest paths between s and every node in the set $V(\mathbf{X}_{unc})$. Denote by $cost(v)$ the shortest distance between node v and the sink s in G'.

The goal of Step 7 is to select among the nodes that cover intervals in $\mathbf{X}_{unc}$, a node v_{best} for which the nodes on the most reliable (s, v_{best})-path is worthy of being included in C_{new}. Such node exists if $\mathbf{X}_{unc} \neq \emptyset$, and C is not a cutset.

Our second observation is that a node v with a highly reliable path to s that covers multiple $\mathbf{X}_{unc}$ intervals should be given higher priority of being selected as v_{best} over similar nodes that cover fewer intervals. To this end, we associate with each node v, a *cost-effectiveness* value $\alpha(v) = \frac{cost(v)}{|covered\_by(v, \mathbf{X}_{unc})|}$. The algorithm favours nodes with small $\alpha(.)$ values.

Our choice of this particular cost-effectiveness function stems from the similarity of our protection interval covering problem and the weighted *set cover* problem (see, e.g., [51]). For this latter problem, using α leads to an approximation algorithm with a logarithmic approximation ratio.

In Step 9, the algorithm adds the nodes of P_{best} (except s and other nodes currently in $C \bigcup C_{new}$) to C_{new} as operating nodes.

Example. Fig. 4.3 illustrates an instance $(G, p, \mathbf{X}, C)$ of the E2P problem where solid lines show network links used by some routing algorithm, node operating probabilities appear on the top left of each node, $\mathbf{X} = \{X_1 = (12, 13, 14, 15), X_2 = (13, 14, 15, 11, 7)\}$, and the input configuration $C = \{(5, fail), (10, op)\}$. All non-sink nodes not in C are free nodes that can be assigned states in C_{new}.

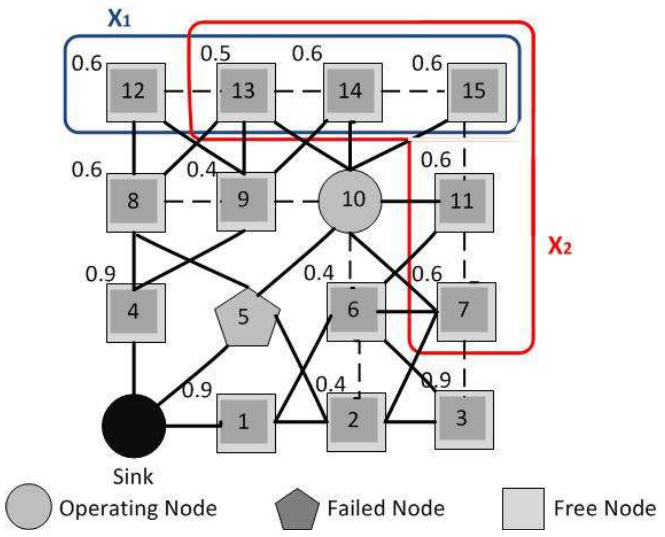


Figure 4.3: Example of extension to a pathset for the BPDREL problem

The algorithm computes for each node v a most reliable (s, v)-path $P_{s,v}$, as well as the $cost(v)$ and $\alpha(v)$ values. For example:

- For node 13, $P_{s,13} = (13, 8, 4, s)$, and $cost(13) = -\log \Pr(P_{s,13}) = w(13) + w(8) + w(4)$. Here, $|covered\_by(13, \mathbf{X}_{unc})| = |\{X_1, X_2\}| = 2$, and $\alpha(13) = cost(13)/2$.

- For node 14, $P_{s,14} = (14, 9, 4, s)$, $cost(14) = -\log \Pr(P_{s,14}) = w(14) + w(9) + w(4)$, and $\alpha(14) = cost(14)/2$.

Since $\alpha(13) < \alpha(14)$, node 13 is preferred over node 14. In fact, node 13 has the least α value among all nodes in $V(\mathbf{X}_{unc})$, and hence it is chosen as v_{best}. The function then augments C_{new} with $\{(13, op), (8, op), (4, op)\}$. ■

4.3.3 Discussion

We remark that function E2P does not give a false negative. That is, it does not return -1 when the input configuration C is extensible to a pathset. This follows since each iteration of the function simply changes some free nodes to operating nodes in Step 9.

4.4 Extension to a Cutset

In this section, we define the optimal *extension to a cutset* (E2C) problem as follows. Given an instance of the BPDREL problem, and a configuration C that is not a pathset, extend C to a cutset $C \cup C_{new}$ so that the occurrence probability $\Pr(C \cup C_{new})$ is as high as possible (i.e., C_{new} has nodes with high failure probabilities). In the definition, C and C_{new} contain two disjoint sets of nodes. Our main contribution in this section is to show that the optimal E2C problem can be solved exactly by repeatedly solving instances of the well known maximum flow problem.

The particular version of the maximum flow problem useful to us has instances of the form (G, s, t, cap) where G is an undirected graph with two distinguished terminal nodes, denoted s and t, and each node $x \in V(G)$ has a real capacity, denoted $cap(x)$. The problem calls for finding the maximum (s,t)-flow amount subject to standard constraints on node capacities and flow conservation. A solver for the maximum flow problem also returns an (s,t)-cut (i.e., a subset of nodes whose removal disconnects s from t) with minimum possible total capacity. Algorithms for solving the above problem where capacities are associated with nodes can be obtained by transforming the problem into one where capacities are associated with directed links. Existing algorithms for solving the tarnsformed problem can then be used (see, e.g., [16]).

To present the result, we need the following definition that refers to any input configuration C, a possible extension configuration C_{new}, and any protection interval X_i.

Definition. The $logq\_capcity(G, C, X_i)$ is a graph G' obtained from G by first deleting all nodes that are assigned a failed state in C. Second, we add a new node, denoted t, and make it adjacent to every node in interval X_i. The two terminals of G' are the sink node s and t (the new node). Third, for each node v in G', $s \neq v \neq t$, that is not assigned an operating state in C and has $q(v) > 0$ set $cap(v) = -\log q(v)$. We then let $maxcap = 1 + \sum_{(v:q(v)>0)} cap(v)$. Finally, for each node v, $s \neq v \neq t$, with either $q(v) = 0$, or v is assigned an operating state in

C, set $cap(v) = maxcap$. ■

Note. To construct the graph G' and solve the maximum flow problem mentioned above in polynomial time (while avoiding the complexity of computing logarithms), we assume the following:

1. all node operating (and failure) probabilities are rational numbers.

2. Addition, subtraction, and comparison operations on flow values required by the maximum flow algorithm are done by performing suitable operations on arguments of logarithms (e.g., adding two flows of magnitude $-\log a$ and $-\log b$ is done by computing the product ab, and dealing with the result as $-\log ab$).

4.4.1 Dealing with a Single Protection Interval

We start by presenting the following results.

Lemma 4.2 *Let $(G, p, \mathbf{X} = \{X_i\}, C)$ be an instance of the E2C problem where C is a configuration of G that is not an X_i-pathset. Computing an optimal extension for C to an X_i-cutset can be done by solving a single instance of the maximum flow problem.*

Proof. Let $(G, p, \mathbf{X} = \{X_i\}, C)$ be as in the lemma, we construct a flow graph $G' = logq\_capcity(G, C, X_i)$. If the input configuration C is not an X_i-pathset then a maximum flow solver returns a minimum capacity (s, t)-cut, denoted V_{cut}, that does not include any node v with $cap(v) = maxcap$ (i.e., no node v is assigned an operating state in C). Denote by C_{cut} the configuration obtained by assigning the failed sate to each node in V_{cut}. We remark that

$$\begin{aligned} -\log \Pr(C_{cut}) &= -\log \textstyle\prod_{v \in V_{cut}} q(v) \\ &= -\textstyle\sum_{v \in V_{cut}} \log q(v) = cap(V_{cut}). \end{aligned}$$

Since the solver identifies an (s, t)-cut V_{cut} with the smallest possible total capacity $cap(V_{cut})$, it then follows that C_{cut} is an extension configuration with the highest possible occurrence probability $\Pr(C_{cut})$, and thus $C \bigcup C_{cut}$ is an optimal solution to the problem. ■

4.4.2 Dealing with Multiple Protection Intervals

Theorem 4.3 *Let $(G, p, \mathbf{X}, C)$ be an instance of the E2C where the configuration C is not a pathset. Computing an optimal E2C solution can be done by solving at most $|\mathbf{X}|$ instances of the maximum flow problem.*

Proof. By definition, for a reduced set of protection intervals, a configuration $C \bigcup C_{new}$ is a cutset iff the failed nodes forbid the covering of some protection interval in $\mathbf{X}$. By Lemma 4.2, for each interval $X_i \in \mathbf{X}$, an optimal X_i-cutset can be computed by solving a maximum flow problem instance. Setting C_{new} to an extension with a highest occurrence probability then gives an optimal solution to the E2C problem. ■

Algorithm 4: Function $\mathrm{E2C}(G, p, \mathbf{X}, C, C_{new})$

```
Input: An instance $(G, p, X, C)$ of the $E2C$ problem
Output: If the E2C problem has a solution return $+1$, and a solution in
        $C \cup C_{new}$. Else (if C is a pathset), return $-1$.
1  if($C$ is a pathset) return $-1$
2  Set $C_{new} = \phi$
3  foreach (interval $X_i \in \mathbf{X}$) do
4      Set the flow network $(G', s, t, cap) = logq\_capcity(G, C, X_i)$
5      Find a minimum $(s,t)$-cut $V_{cut}$
6      if (no $(s,t)$-flow exists) then
           reset $C_{new} = \emptyset$; return $+1$
7      else if ($V_{cut}$ has at least one node $v$ with $cap(v) = maxcap$) then
           continue
8      else
           update $C_{new}$ with the best found $V_{cut}$
       end
   end
9  return $+1$
```

Function $E2C$ presents the overall structure of the algorithm that implements the above observations. Step 1 returns with failure (-1) if C is a pathset. Step 2 initializes $C_{new} = \emptyset$. The main loop in Step 3 iterates over each interval $X_i \in \mathbf{X}$. Steps 4 and 5 compute a minimum cut in the constructed flow graph for interval X_i. If C is already an X_i-cutset then Step 6 returns with success ($+1$). Else, if the found cut contains one of the nodes assigned the operating state then Step 7 continues the loop to process the next interval in $\mathbf{X}$. On the other hand, if a minimum X_i-cutset composed of free nodes is found then Step 8 updates C_{new} with the best cutset found thus far. Finally, Step 9 returns with success $(+1)$ and the solution in C_{new}.

Example. Fig. 4.3 illustates an instance $(G, p, \mathbf{X}, C)$ of the E2C problem with parameters as explained in the previous example. For intervals X_1 and X_2, the function computes the cutset configurations $C_{cut,1} = C_{cut,2} = \{(2, fail), (6, fail), (8, fail), (9, fail)\}$ (of probability 0.0864). The function then sets $C_{new} = C_{cut,1}$. ■

4.5 The Main Algorithm

In this section we outline the general structure of our algorithm that computes lower bounds (LBs) and upper bounds (UBs) on $Rel(G, p, \mathbf{X})$. To simplify the presentation, we focus on the work done by the algorithm to compute LBs from pathsets using function E2P. Computing UBs is done by replacing pathsets with cutsets, and calling function E2C instead of function E2C.

Similar to function Main in Section 3.5, the algorithm is organized around a main loop that iterates a user specified number of times. Throughout execution, the algorithm maintains a table that stores a set of s-disjoint configurations. In particular, each *valid* entry of the table contains a configuration C, an extension C_{new} obtained by calling function E2P, and the corresponding occurrence probability $\Pr(C \bigcup C_{new})$. If no such extension is possible, we mark the entry as *invalid*.

Initially, the table contains only one entry corresponding to the empty configuration $C = \emptyset$, and one of its possible high probability extensions. Each iteration starts by selecting a valid entry (C, C_{new}) with the highest occurrence probability and generates a set of configurations derived from the base configuration C. Again, similar to function Main (Section 3.5), the derivation process is based on repeated application of a *factoring* theorem.

The process basically assigns states to nodes in prefixes of C_{new}, and augments the base configuration C with the generated node-state pairs. The process used is similar in principle to the process used, e.g., in [2]. For each generated configuration C', the algorithm computes (if possible) a corresponding extension C'_{new} and the probability $\Pr(C' \bigcup C'_{new})$, and stores the entry in the table. So, at the end of the iteration, the table contains both the selected pathset (C, C_{new}), and the derived configurations.

The main loop terminates when either all user specified iterations are exhausted, or all valid entries are processed. In the latter case, the algorithm terminates when no more work remains to be done; here, the algorithm guarantees that the computed LB (or UB) is the exact solution. An important aspect of the implementation correctness is that, upon termination, all entries in the table correspond to pairwise s-disjoint configurations.

In summary, **(a)** the algorithm exhaustively generates a maximal set of pairwise s-disjoint configurations, **(b)** it avoids generating all network states by invalidating configurations that can not be extended to pathsets, and **(c)** it makes efficient use of the user specified number of iterations by targeting the generation of the best pathset stored in its information base in each iteration.

4.6 Numerical Results

In this section, we explore some of the important aspects and potential uses of our bounding algorithms using simulation. We run a number of experiments both on well structured graphs as well as random graphs. In particular, to allow comparison with the exact results obtained in [46], we experiment with grid networks. Below, we present selected results of both types.

To start, we adopt the following notation: a $W{\times}L$ grid has W rows at (x, y)-coordinates $y = 0, 1, 2, \ldots, W-1$, and L columns at coordinates $x = 0, 1, 2, \ldots, L-1$. In [46], the sink node is positioned at the bottom left corner of the grid at coordinates $(0, 0)$. We also experiment with sink nodes placed in the interior of the network.

Some routing algorithms restrict the routes to the sink to shortest paths (i.e., paths with fewest number of hops), while others utilize all of the available links.

We report results when using both types of routes: *BFS-links* (Breadth First Search links) routing utilizes shortest paths, while *all-links* routing utilizes all available links.

4.6.1 Exact Computations

Our main algorithm avoids the generation of all network states by discarding configurations that can not be extended to pathsets (or cutsets) at all stages. The reduction in the generated configurations allows exact computations on networks of different sizes. All results reported in table 4.1 runs in less than 4 minutes on a personal laptop computer using a Matlab implementation. As can be seen, the number of generated configurations to get exact results is significantly less than the number of network states in each case.

Table 4.1: Exact computations for the BPDREL problem

$W{\times}L$	**Network states**	**Generated configuration**
3×3	2^8	11
4×4	2^{15}	65
5×5	2^{24}	721
6×6	2^{35}	16389

4.6.2 Gaps between Bounds and Exact Solutions

Here, we comment on the gaps between the obtained bounds and the exact results in [46]. We use a $W{\times}W$ grid network, $W \in [2, 10]$, with down-left diagonal links ($= \{((x, y), (x-1, y-1) : x, y > 0\}$) where the sink node is located at the bottom left corner. All nodes have equal operation probability $p = 0.5$. The BPDREL problem has one pair of entry-exit sides that split the perimeter into two intervals: interval X_{top} contains nodes in the top row, and $\overline{X}_{top}$ contains the remaining perimeter nodes. We performed 1000 iterations. The results show that the gap between the bounds and the exact solution increases with increasing network size. This is expected as larger networks have more configurations to consider that can not be exhausted in the few iterations performed. We also note that the computed LB tends to be a more accurate estimate of the exact result than the computed UB. This can be explained by the relative small size (and hence higher occurrence probability) of pathsets compared to cutsets.

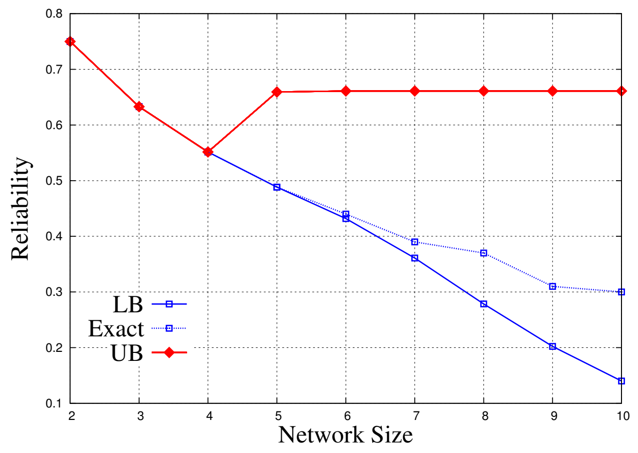


Figure 4.4: Effect of network size on the obtained bounds for the BPDREL problem

4.6.3 Sink Placement Optimization

Given a set $\mathbf{X}$ of protection intervals, the question of where to place the sink node so as to maximize $Rel(G, p, \mathbf{X})$ is an interesting WSN design problem. Our bounding algorithms provide a tool for tackling the problem. Our results address this design problem when the network G is a square $W \times W$ grid for $W = 6$, and also when G is a random network. Note that the algorithm presented in [46], although exact for grid networks, it does not deal with sink nodes placed in the interior of the network. The network has a perimeter of 20 nodes. For a sink placed in the interior of the grid, the set $\mathbf{X}$ has two protection intervals with nodes at the following (x, y)-coordinates: $X_{top} = ((0,5), (1,5), \dots, (5,5))$, so $|X_{top}| = 6$ nodes, and $\overline{X}_{top}$ has the remaining 14 perimeter nodes. We vary the sink location on the main grid diagonal (i.e., nodes at coordinates $(0,0)$, $(1,1)$, . . ., $(5,5)$).

Varying the routing method. Fig. 4.5 shows the obtained results when varying the sink location and the routing method. Both of the BFS-links and all-links routing curves show similar behaviour indicating a preference to place the sink node at coordinates $(4,4)$. To analyze the situation, we counted for each possible sink location, the number of pathsets of each possible size (not counting the sink in a pathset). The findings indicate that node $(4, 4)$ tends to have more pathsets of small sizes than other diagonal nodes, although, e.g., node $(0, 0)$ has a pathset of the smallest possible size of one node. Such small sized pathsets have relatively high occurrence probability when all nodes have equal operating probability.

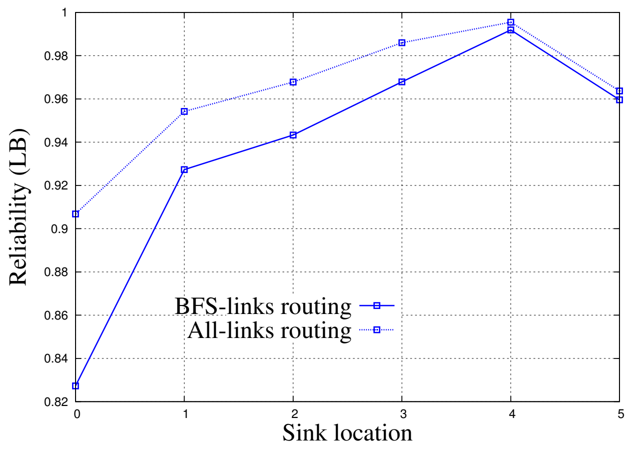


Figure 4.5: Effect of varying sink location on $Rel(G)$ of the BPDREL problem for different routing methods ($p = 0.7$)

Random networks. Fig. 4.6 shows an instance of the sink placement problem on a random network deployed in a field of 7×7 units. Node operation probabilities are also assigned randomly. Each node is assumed to have a transmission range of 2 units. The BPDREL problem under consideration has one entry-exit pair of sides, as shown. The two sides split the perimeter into two intervals, denoted X_1 and X_2, where $|X_1| = 6$ nodes, and $|X_2| = 24$ nodes.

Six locations (shown as rectangles) are considered for sink placement. Based on the obtained results, location 5 is superior to others. To gain insight into the situation, we counted the number of pathsets of each possible size for each sink location. The findings can be summarized as follows. Placing the sink in the interior of the network (e.g., locations 1 through 4) requires pathsets to cover both X_1 and X_2. Using a transmission range of 2 units, each such pathset requires at least 3 nodes (not counting the sink). On the other hand, placing the sink on the perimeter (e.g., locations 5 and 6) requires the pathsets to cover only one interval (either X_1 for location 6, or X_2 for location 5). Such pathset requires 2 (or more) nodes. But interval X_2 is much longer than X_1. Consequently, it offers the construction of more pathsets. Thus, there is a potential benefit of locating the sink at location 5.

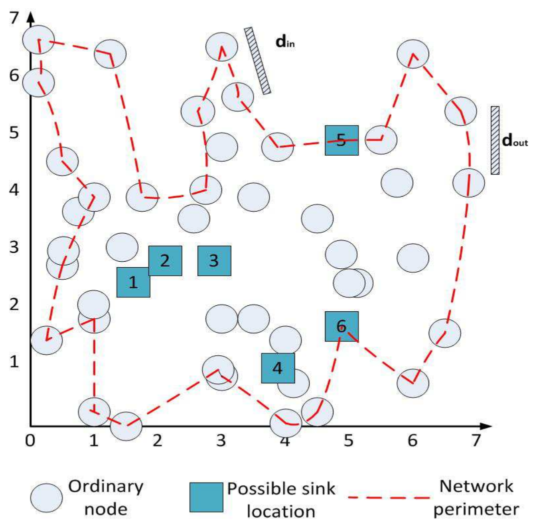


Figure 4.6: An instance of a random network (BFS-links routing) for the BPDREL problem

4.7 Concluding Remarks

In this chapter, we develop efficient bounding algorithms for a WSN area surveillance problem where nodes can fail in sensing and/or communication with given probabilities. The algorithm works on any WSN network where the sink is located anywhere in the network. Using our devised algorithms, we have investigated an optimal sink location problem to maximize the network's overall reliability. This illustrates a potential use of our methods in tackling a complex WSN design problem.

Chapter 5

Breach Path Reliability for Directional WSNs

Wireless Sensor Networks (WSNs) equipped with directional communication and sensing devices provide a high level of tunability needed in optimizing their performance in critical applications. In this chapter, we formalize the directional breach path detection reliability problem ($\mathrm{DIR\text{-}BPDREL}$) that quantifies the ability of such networks to jointly detect and report unauthorized traversal through a network when communication and sensing devices fail independently of each other. Our main contributions in this chapter are as follows:

1. We extend the unified framework for computing lower and upper bounds used in the previous two chapters to the current problem. This is done by examining the specific details of the extension to pathset and cutset problems in the context of the $\mathrm{DIR\text{-}BPDREL}$ problem.
2. Our simulation results show the effectiveness of our devised tools in tackling an example network design problem.

Some of the results in this chapter appear in [22].

5.1 System Model

In this section, we introduce the needed concepts and notation to explain our bounding approach and algorithms. Omnidirectional versions of these concepts appear in [46] and chapter 4.

5.1.1 Network and Directionality Model

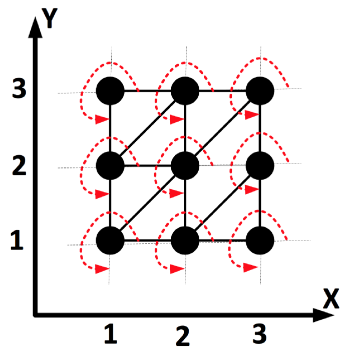
Throughout the chapter, a WSN is modelled by an undirected communication graph $G_{com} = (V \bigcup \{s\}, E_{com})$, and a sensing graph $G_{sense} = (V \bigcup \{s\}, E_{sense})$ on a set V of directional sensor nodes, and a distinguished sink node s. The adoption of two graphs in the model generalizes the single graph model used previously in chapter 4.

In G_{com}, a link (x, y) exists if both nodes can reach each other. In G_{sense}, a link (x, y) exists if every point on the line segment (x, y) can be sensed by node x and/or node y. This latter property is needed to guarantee detection of any crossing of the line segment (x, y).

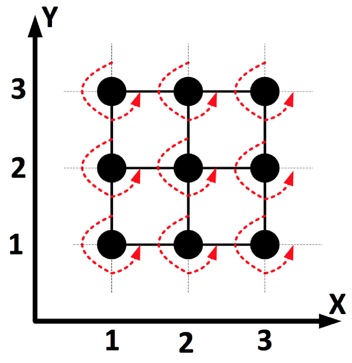
The exact links in G_{com} and G_{sense} are determined by directionality parameters, denoted $DIR_{com}(x)$ and $DIR_{sense}(x)$, respectively, associated with each node x. In each case, directionality is specified by a triple, denoted $(r, \theta_{mid} \pm \alpha)$, where

- r is a transmission/reception radius (either communication or sensing),
- $\theta_{mid} - \alpha$ specifies a counterclockwise (CCW) angle between two rays emanating from x; a horizontal ray that extends to the right, and a ray that defines the **start** of the boundary of the transmission/reception region, and
- $\theta_{mid} + \alpha$ specifies the CCW angle between the horizontal ray, and a second ray that defines the **end** of the boundary of the transmission/reception region.

Example. Fig. 5.1a illustrates G_{com} of a subnetwork composed of 3x3 grid nodes with one unit of horizontal (or vertical) spacing. When $DIR_{com} = (1.7, 135° \pm 135°)$, the transmission/reception region for each node is bounded by two rays emanating from the node with CCW angles $(\theta_a = 0°, \theta_b = 270°)$, as shown by the dashed arrows. A link (x, y) exists in G_{com} if $dist(x, y) \leq 1.7$ unit.



(a) G_{com}



(b) G_{sense}

Figure 5.1: Example of directional parameters

Fig. 5.1b illustrates G_{sense} of a subnetwork on the same set of nodes. When $DIR_{sense} = (0.5, 225° \pm 135°)$, the sensing region for each node is bounded by two rays emanating from the node with CCW angles $(\theta_a = 90°, \theta_b = 360°)$, as shown by the dashed arrows. A link (x, y) exists in G_{sense} if each point on the line segment (x, y) is covered by x and/or y. Here, since $r = 0.5$, both end nodes of a link are needed to cover the link. ■

The above directionality model allows for achieving spatial selectivity while controlling the consumed power through suitable setting of the transmission/reception radius r as a function of the offset angle α. For example, in Section 5.6, we experiment with setting $r = \frac{360°}{2 \cdot \alpha}$ to capture the above effect.

5.1.2 Breach Path Model

We fix an embedding of G_{sense} in the plane so that the network's perimeter corresponds to a polygon with well-defined set of perimeter nodes, denoted X, and polygon links, called *sides*. An intruder (breach) path crossing G_{sense} is associated with a pair (d_i^{in}, d_i^{out}) of entry-exit sides on the polygon. Entry-exit sides can be exchanged, so a breach path is undirected. Without loss of generality, we assume that each breach path crosses (or touches) only two perimeter sides of the network.

Our problem deals with a given set of entry-exit pairs of sides $D = \{(d_i^{in}, d_i^{out} : i = 1, 2, \cdots\}$, $d_i^{in} \neq d_i^{out}$. A (d_i^{in}, d_i^{out})-attack is any breach path that uses the specified sides. A D-attack is any breach path that uses any pair of sides in D.

5.1.3 Reliability Model

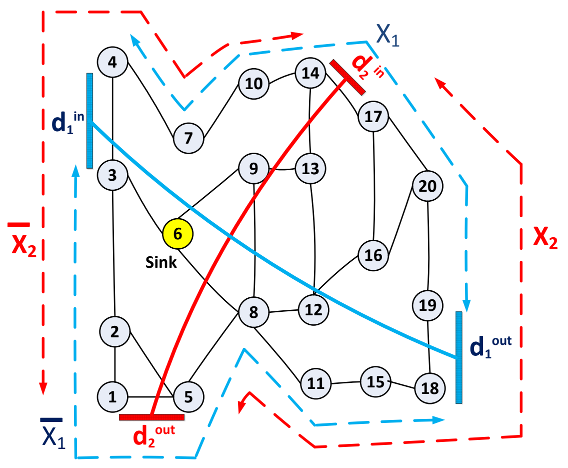
For any non-sink node x, we adopt a model with independent operation probabilities for the communication and the sensing devices, denoted $p_{com}(x)$, and $p_{sense}(x)$, respectively. The sink is assumed to be perfectly reliable. Thus, each non-sink node x can be in one of 3 possible states with the following probabilities: $p_{cs}(x) = p_{com}(x) \times p_{sense}(x)$: the probability that x can perform both communication and sensing, $p_{cns}(x) = p_{com}(x) \times (1 - p_{sense}(x))$: the probability that x can communicate but not sense, and $p_{fail}(()x) = 1 - p_{cs}(x) - p_{cns}(x)$: the probability that x can not communicate with any other node. We assume that the network utilizes adaptive routing that can utilize any existing operational path.

A random failure event leaves a network on n nodes in one of $O(3^{n-1})$ possible states (recall that the sink is reliable). Each state $T = (T_{cs}, T_{cns})$ is defined by a subset T_{cs} of nodes that can communicate and sense, and another disjoint subset T_{cns} of nodes that can communicate but not sense. Each of the remaining nodes $T_{fail} = T \setminus \{T_{cs} \bigcup T_{cns})$ is failed. The probability of being in state T, denoted $\Pr(T)$, is the product of the p_{cs}, p_{cns}, and p_{fail} probabilities of nodes in T_{cs}, T_{cns}, and T_{fail}, respectively.

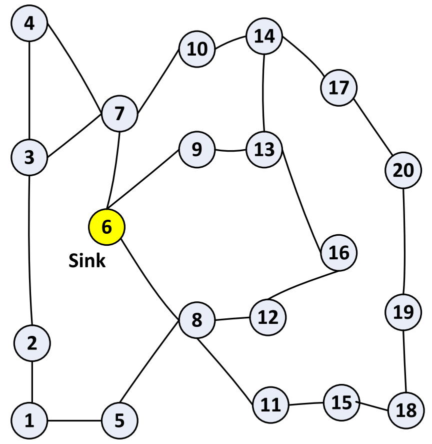
A state T is *operating* if it guarantees the joint detection and reporting to the sink of any D-attack. The reliability of the network is the sum $\sum \Pr(T)$ over all operating network states T in (G_{com}, G_{sense}). We now define our problem:

Definition (the $\mathrm{DIR\text{-}BPDREL}$ **problem):** Let (G_{com}, G_{sense}) be a WSN model where each node x has operating probabilities $p_{com}(x)$ and $p_{sense}(x)$, and directional parameters $DIR_{com}(x)$ and $DIR_{sense}(x)$. In addition, let $D = \{(d_i^{in}, d_i^{out}) : i = 1, 2, \cdots\}$, $d_i^{in} \neq d_i^{out}$, be a set of entry-exit nodes on the perimeter polygon of an embedding of G_{sense}. Find the probability $Rel(G_{com}, G_{sense}, p, D)$ (or, $Rel(G_{com}, G_{sense})$ for short) that the network is in a state that ensures joint detection and reporting of any D-attack. ■

We recall from Theorem 4.1 that the omnidirectional BPDREL is #P-hard even when restricted to a wireless grid network. Consequently, the more general $\mathrm{DIR\text{-}BPDREL}$ problem is #P-hard.



(a) An example G_{sense} of a WSN



(b) An example G_{com} of a WSN

Figure 5.2: An instance of the DIR-BPDREL problem

5.2 Protection Intervals, Pathsets, and Cutsets

This section extends the concepts of protection intervals, network configurations, pathsets and cutsets discussed, e.g., in Chapter 4 to our current problem. The concepts are needed to present our bounding approach.

5.2.1 Protection Intervals

Let $(G_{com}, G_{sense}, p, D)$ be an instance of the DIR-BPDREL problem, $X \subseteq V$ be the set of perimeter nodes in G_{sense}, and $(d^{in}, d^{out}) \in D$ be an entry-exit pair of sides. Similar to the development in Chapter 4, the *protection intervals* of (d^{in}, d^{out}) are obtained by traversing the perimeter nodes X in a clockwise direction, and writing X as two node-disjoint sequences:

$$X_1 = (second(d^{in}), x_i, x_{i+1}, \ldots, first(d^{out})), \text{ and}$$
$$X_2 = (second(d^{out}), x_j, x_{j+1}, \ldots, first(d^{in})).$$

where $first(d)$ and $second(d)$ denote the two end points of a side d.

Example. In Fig. 5.2a, we have $(d_1^{in} = (3,4), d_1^{out} = (19,18))$, and $(d_2^{in} = (14,17), d_2^{out} = (5,1))$. The corresponding protection intervals $(X_1, \overline{X_1})$ and $(X_2, \overline{X_2})$ are as shown in the figure. ■

We denote by $\mathbf{X}$ the set of pairs of protection intervals associated with the set D of entry-exit pairs of sides. We also note that given a protection interval X, one can immediately deduce the associated entry-exit pair of sides (d^{in}, d^{out}) on the polygon surrounding G_{sense}, and the other associated protection interval $\overline{X}$.

The importance of protection intervals arises since a path in G_{sense} where each node is in the state cs, and the path connects some node on X to some node on $\overline{X}$ is both sufficient and necessary to intersect every possible breach path between the corresponding entry-exit sides. Thus, any operating network state T of a given WSN contains at least one such *sensing barrier path* for each pair of protection intervals in $\mathbf{X}$.

Since $\mathbf{X}$ suffices to identify D in any problem instance, we henceforth use the notation $(G_{com}, G_{sense}, p, \mathbf{X})$ to refer to a given instance of the DIR-BPDREL problem.

5.2.2 Network Configurations

A network *configuration* C is a partial network state; that is C assigns to some nodes (but not necessarily all) one of the possible states in the set $\{cs, cns, fail\}$. The remaining nodes are *free* in C. We use $C(x)$ to denote the state s_x of node x in C.

We also use the concept of statistical disjointedness: two configurations C_1 and C_2 are **s-disjoint** if at least one node that occurs in both configurations is assigned different states in the configurations. Thus, $\Pr(C_1 \cup C_2) = \Pr(C_1) + \Pr(C_2)$.

5.2.3 Pathset Structures

Given an instance $(G_{sense}, G_{com}, , p, \mathbf{X})$ of the DIR-BPDREL problem, a pathset of the instance is any operating configuration C. A fundamental problem in our approach is to find pathsets that have high occurrence probability $\Pr(C)$ (subject to some additional constraints).

In this section, we identify a key structural property of pathsets that enable us to efficiently find desired pathsets. The property can be stated as follows.

Lemma 5.1 *A configuration C is a pathset of $(G_{sense}, G_{com}, p, \mathbf{X})$ if and only if for each pair of intervals $(X_i, \overline{X_i}) \in \mathbf{X}$ we have*

a) G_{sense} has a path, denoted $P_{sense}(X_i, \overline{X_i})$, connecting some node $x \in X_i$ to some node $\overline{x} \in \overline{X_i}$, where each node in the path is assigned the state cs in configuration C, and

b) each node in $P_{sense}(X_i, \overline{X_i})$ can reach the sink in G_{com} by a path composed of nodes assigned either state cs or cns in configuration C. ■

The rationale behind the above lemma is that condition (a) guarantees that each path $P_{sense}(X_i, \overline{X_i})$ can detect any (d_i^{in}, d_i^{out})-attack. Moreover, condition (b) guarantees that the detected event can be reported to the sink. Thus, C is a pathset, as required.

Example. In G_{sense} of Fig. 5.2a, the following paths satisfy condition (a) above:

- $P_{sense}(X_1, \overline{X_1})$: the edge $((3, cs), (4, cs))$
- $P_{sense}(X_2, \overline{X_2})$: the edge $((14, cs), (17, cs))$

Hence, each path forms a sensing barrier path. In G_{com} of Fig. 5.2b, the following additional node-state pairs guarantee that each node in each of the above two sensing barrier paths can reach the sink at node 6: $\{(7, cns), (10, cns)\}$. Thus, configuration $C = \{(3, cs), (4, cs), (7, cns), (14, cs), (17, cs), (10, cns)\}$ is a pathset. ■

5.2.4 Cutset Structures

Given an instance $(G_{sense}, G_{com}, , p, \mathbf{X})$ of the DIR-BPDREL problem, a cutset of the instance is a configuration C that can **not** be extended to a pathset even if all free nodes are assigned the cs state. For a given pair $(X, \overline{X}) \in \mathbf{X}$, we also define an $(X, \overline{X})$-cutset to be a configuration that can not be extended so as to guard against any $(X, \overline{X})$-attack. Thus, a necessary and sufficient condition for C to be a cutset is that it includes an $(X, \overline{X})$-cutset for at least one pair of protection intervals $(X, \overline{X}) \in \mathbf{X}$.

Similar to the previous section, a fundamental problem in our approach for computing UBs is to find cutsets that have high occurrence probability. The following observation specifies a structural property that is used to find good quality cutsets.

Lemma 5.2 *A configuration C is an $(X, \overline{X})$-cutset of $(G_{com}, G_{sense}, p, \mathbf{X})$ if there exist two subsets of nodes, denoted $U_{cns,fail}$ and U_{fail}, such that each node $x \in U_{cns,fail}$ is assigned a state $C(x) \in \{cns, fail\}$, and each node $x \in U_{fail}$ is assigned a state $C(x) = fail$, and any possible path $P_{sense}(X, \overline{X})$ in G_{sense} that connects some node in X to some node in $\overline{X}$ has either*

a) at least one node in $U_{cns,fail}$, or

b) the failed nodes in U_{fail} disconnect at least one node on $P_{sense}(X, \overline{X})$ from the sink node. ■

The rational behind the lemma is that condition (a) implies that the path $P_{sense}(X, \overline{X})$ is not an operational sensing barrier path since it has at least one non-sensing node. Likewise, condition (b) implies that at least one node on such a path can not report a detected intrusion event.

Example. In G_{sense} of Fig. 5.2a, the two sets ($U_{cns,fail} = \{3, 9, 12, 18\}$,$U_{fail} = \emptyset$) form an $(X_1, \overline{X_1})$-cutset that forbids the construction of a sensing barrier path $P_{sense}(X_1, \overline{X_1})$.

In G_{com} of Fig. 5.2b, the two sets ($U_{cns,fail} = \emptyset$,$U_{fail} = \{7, 8, 9\}$) disconnects the sink, and thus they form an $(X_1, \overline{X_1})$-cutset that disconnects the sink from some node on any possible sensing barrier path $P_{sense}(X_1, \overline{X_1})$, as required by condition (b). ■

In the above example, each pair $(U_{cns,fail}, U_{fail})$ satisfies either condition (a) or condition (b) for any possible path $P_{sense}(X_1, \overline{X_1})$. The lemma, however, takes care of more complex scenarios where the pair $(U_{cns,fail}, U_{fail})$ satisfies condition (a) for some paths $P_{sense}(X_1, \overline{X_1})$, and condition (b) for the remaining $P_{sense}(X_1, \overline{X_1})$ paths.

5.3 Overview of the Bounding Approach

Our approach obtains lower bounds (LBs) on $Rel(G_{com}, G_{sense})$ by computing a set of pairwise s-disjoint pathsets $\{P_1, P_2, \cdots, P_r\}$, and observing that $Rel(G) \geq \sum_{i=1}^{r} \Pr(P_i)$ where the summation on the RHS constitutes the intended lower bound.

In a parallel vein, upper bounds (UBs) are obtained by computing a set of pairwise s-disjoint cutsets $\{U_1, U_2, \cdots, U_r\}$, and observing that $Rel(G) \leq 1 - \sum_{i=1}^{r} \Pr(U_i)$ where the RHS is the desired upper bound.

As done in Chapters 3 and 4, we implement the above approach with an iterative algorithm that performs a user specified number of iterations while maintaining a set of s-disjoint configurations. In each iteration, the algorithm selects an unprocessed configuration C and seeks to extend it either to a pathset (in LB computations), or a cutset (in UB computations). The new extension, denoted C_{new}, is desired to have a high occurrence probability $\Pr(C)$. The extension process is critical to the quality of the obtained bounds. Configurations that can not be extended are ignored. An exact solution is obtained if the algorithm processes all of its generated configurations. In the next sections, we discuss in more detail the extension to a pathset problem and the extension to a cutset problem.

5.4 Optimal Extension to a Pathset

In this section, we examine the optimal *extension to a pathset* (E2P) problem that calls for computing a configuration C_{new} such that $C \bigcup C_{new}$ is a pathset, and $\Pr(C_{new})$ is as high as possible. The E2P problem is discussed in chapter 4 for the BPDREL problem where we deal with the special case $G_{sense} = G_{com}$.

For the DIR-BPDREL problem, the E2P problem is NP-complete as the E2P problem for the restricted version BPDREL problem is NP-complete. Hence, we consider in this section developing an efficient heuristic algorithm. Our algorithm computes a solution to any given instance of the E2P problem, if one exists. The computed solution, however, may not be an optimal solution. Function E2P outlines the general steps of the algorithm. The function performs its work by calling functions find_SB and connect_SB described below.

Algorithm 5: Function $\mathrm{E2P}(G_{sense}, G_{com}, p, \mathbf{X}, C)$

```
Input: An instance $(G_{sense}, G_{com}, p, \mathbf{X}, C)$ of the $E2P$ problem
Output: If the $E2P$ has a solution, return a configuration $C_{new}$ containing
        additional nodes used to form a pathset. Else, return an error value.
1 call function $\mathrm{find\_SB}(G_{sense}, G_{com}, p, \mathbf{X}, C)$ to identify a set of sensing
    barrier paths that satisfies condition (a) in Lemma 5.1. $C_{new\_sense}$ is the
    configuration of the new added nodes in the set of paths.
2 call function $\mathrm{connect\_SB}(G_{com}, p, \mathbf{X}, C, C_{new\_sense})$ to identify a set of
    nodes that satisfies condition (b) in Lemma 5.1; that is, a set of nodes that
    connect every node in the set of sensing barrier paths found in step 1 to the
    sink. $C_{new\_com}$ is the configuration of the new added nodes in this set.
3 return $C_{new} = C_{new\_sense} \bigcup C_{new\_com}$
```

5.4.1 Use of Most Probable Paths

To obtain good solutions, function E2P makes frequent use of an algorithm to solve the following problem. Given a weighted undirected graph where each node x has an associated weight $\tilde{p}(x)$, and two disjoint subsets of nodes X and Y of G, find the *most probable* path that links some node in X to some node in Y. The weight $\tilde{p}(x)$ may either be zero, indicating that x is to be deleted from G before the computations, or a positive probability value.

An efficient algorithm for solving the problem is readily available by using a standard shortest path algorithm through the following transformation: **(1)** delete nodes with $\tilde{p}(x) = 0$ from G, **(2)** associate with each node x a non-negative weight $w(x) = -\log \tilde{p}(x)$ (so, a small $\tilde{p}(x)$ probability maps to a large positive weight $w(x)$), and **(3)** introduce two new nodes x and y; connect x (respectively y) to each node in X (respectively, Y). A shortest (x, y)-path in the constructed graph is a most probable path connecting some node in X to some node in Y in the original problem, as required.

The asymptotic running time of the algorithm is the same as the running time of the used shortest path problem, denoted T_{SP}. For the purpose of analyzing our algorithm that solves the E2P problem, we use T_{SP} as a component in our timing analysis.

5.4.2 Finding a Sensing Barrier

Function find_SB aims at computing a set of sensing barrier paths $\{P_{sense}(X_i, \overline{X_i}) : i = 1, 2, \cdots\}$ in G_{sense} that satisfies condition (a) in Lemma 5.1, whenever possible.

```
Algorithm 6: Function find_SB($G_{sense}, G_{com}, p, \mathbf{X}, C$)
Input: An instance $(G_{sense}, G_{com}, p, \mathbf{X}, C)$ of the $E2P$ problem
Output: If the E2P problem has a set of sensing barrier paths satisfying
        condition (a) in Lemma 5.1 then return a configuration $C_{new\_sense}$
        containing the additional nodes used to form the paths. Else, return an
        error value.
1 set $C_{new\_sense} = \emptyset$
2 use $(G_{com}, p, C)$ to assign to each node $x$ a weight $\tilde{p}(x)$ (high values indicate
    node importance)
3 foreach (pair $(X_i, \overline{X_i}) \in \mathbf{X}$) do
4     find a most probable $(X_i, \overline{X_i})$-path in $(G_{sense}, \tilde{p})$. If no such path exists
        return an error value in $C_{new\_sense}$. Else, denote the path by
        $P_{sense}(X_i, \overline{X_i})$.
5     foreach (free node $x$ in $P_{sense}(X_i, \overline{X_i})$) do
6         add $(x, cs)$ to configuration $C_{new\_sense}$
7         set $\tilde{p}(x) = 1$
      end
  end
8 return $C_{new\_sense}$
```

The function returns an error value in C_{new_sense} if no such set exists (no solution is found). On the other hand, if a solution exists then nodes in the solution that are not in C are returned in the new configuration C_{new_sense}. The function aims at maximizing the occurrence probability $\Pr(C_{new})$. To this end, the function performs the following steps.

Step 1 initializes $C_{new\_sense} = \emptyset$. Step 2 initializes a probability array $\tilde{p}(x \in G_{com})$ that guides the function so that **(a)** it never selects a node x that can **not** reach the sink node in (G_{com}, C), and **(b)** it favours the selection of nodes that can jointly be assigned the cs state and reach the sink with high probability. The setting of $\tilde{p}(x)$ is such that higher numerical values indicate higher desirability of selecting node x. More specifically, for each node x, Step 2 initializes $\tilde{p}(x)$ as follows:

1. If $C(x)$ (i.e., x's state in C) is in $\{fail, cns\}$, or if x can not reach the sink

in (G_{com}, C) then set $\tilde{p}(x) = 0$. Nodes in this class can not appear in any sensing barrier path (hence, the setting $\tilde{p}(x) = 0$).

2. Else, if $C(x) = cs$ then set $\tilde{p}(x) = 1$. Nodes in this class are already part of the input configuration; they can contribute to building the required sensing barrier paths (at no extra cost).

3. Else, if $C(x) = free$ then set $\tilde{p}(x) =$ the probability of a most probable path in (G_{com}, C) connecting x to the sink. Nodes in this class are candidates for use in building the required sensing barrier paths. The setting of $\tilde{p}(x)$ gives more suitable nodes higher values.

Steps 3 to 7 iterate over all pairs of protection intervals $(X_i, \overline{X_i}) \in \mathbf{X}$. Each iteration finds a suitable most probable path in $(G_{sense}, \tilde{p})$. Step 7 sets $\tilde{p}(x) = 1$ if x is a free node selected for inclusion in any path.

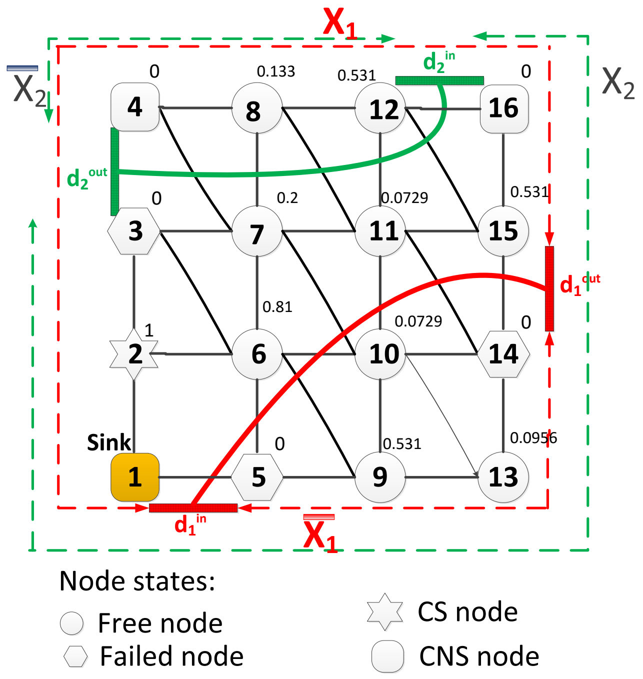


Figure 5.3: An example G_{sense} for function find_SB

Example. Fig. 5.3 illustrates a G_{sense} of a WSN. The shape of each node specifies its state in the input configuration $C = \{(2, cs), (3, fail), (4, cns), (5, fail),$

$(14, fail), (16, cns)\}$. Node 1 is the sink node. Each non-sink node x is labelled with the weight $\tilde{p}(x)$, as defined above. Function find_SB finds the following most probable paths (and assigns to the new nodes the following states):

- $P_{sense}(X_1, \overline{X_1})$: $((2, cs), (6, cs), (9, cs))$
- $P_{sense}(X_2, \overline{X_2})$: $((12, cs), (15, cs))$

The function returns $C_{new\_sense} = \{(6, cs), (9, cs), (12, cs), (15, cs)\}$ ■

5.4.3 Connecting a Sensing Barrier

Function connect_SB (Algorithm 7) aims at adding new nodes to configuration $C \bigcup C_{new\_sense}$ to satisfy condition (b) in Lemma 5.1. That is, we want to ensure that each node on each sensing barrier path can reach the sink. The function does not fail since care has been given in selecting each node on any such path to ensure that the node can reach the sink in (G_{com}, C).

Algorithm 7: Function connect_SB$(G_{com}, p, \mathbf{X}, C, C_{new_sense})$

```
Input: Configuration $C \bigcup C_{new\_sense}$ satisfies condition (a) in Lemma 5.1 for
       each pair of intervals in $\mathbf{X}$.
Output: If each node on each sensing barrier path can be connected to the sink
        to satisfy condition (b) of Lemma 5.1 then return a configuration
        $C_{new\_com}$ containing the additional nodes used. Else, return an error
        value.
1  set $C_{new\_com} = \emptyset$
2  use configuration $C \bigcup C_{new\_sense}$ to assign to each node $x$ a weight $\tilde{p}(x)$
     (high values indicate node importance)
3  set $V_{SBP}$ = nodes of the computed sensing barrier paths
4  while ($V_{SBP} \neq \emptyset$) do
5      Let $P$ = a most probable path linking some node in $V_{SBP}$, say $x_{new}$, to
         the sink node.
6      foreach (free node $x \in free(P)$) do
7          let $s_x$ be the state of $x$ (either $cs$ or $cns$) with higher probability; add
             $(x, s_x)$ to $C_{new\_com}$
8          set $\tilde{p}(x) = 1$
       end
9      remove $x_{new}$ from $V_{SBP}$
   end
10 return $C_{new\_com}$
```

Step 1 initializes $C_{new\_com} = \emptyset$. Step 2 initializes the array $\tilde{p}(x \in G_{com})$ that guides the function to favour selection of nodes that are either already assigned a state in $\{cs, cns\}$, or a free node that has a high $\max(p_{cs}(x), p_{cns}(x))$ probability.

In more detail, Step 2 initializes $\tilde{p}(x)$ as follows:

1. If $C(x)$ (x's state in C) is $fail$ then set $\tilde{p}(x) = 0$.

2. Else, if x's state in $C \bigcup C_{new\_sense}$ is in $\{cs, cns\}$ then set $\tilde{p}(x) = 1$.

3. Else, if x is a free node in C then set $\tilde{p}(x) = \max(p_{cs}(x), p_{cns}(x))$.

Step 3 sets V_{SBP} to nodes in the computed sensing barrier paths; each node in the set needs to be connected to the sink in the final solution of the E2P problem. Steps 4 to 9 iterate until all nodes in V_{SBP} are connected to the sink. Step 5 finds a most probable path P in $(G_{com}, \tilde{p})$ that links some node, denoted x_{new}, to the sink. Steps 6 to 8 assign a state to each free node x in P, and adds the node-state pair (x, s_x) to C_{new_com}, where state $s_x \in \{cs, cns\}$ is the state of x with higher p_{s_x} probability. Step 8 updates $\tilde{p}(x)$ to 1.

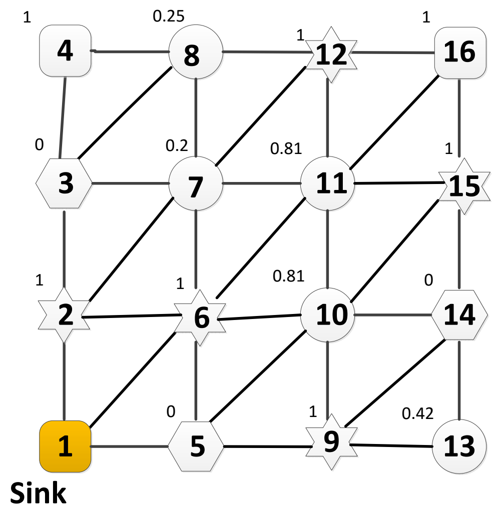


Figure 5.4: An example G_{com} for function connect_SB

Example. Fig. 5.4 illustrates G_{com} for the network in Fig. 5.3. The shape of each node in the diagram specifies its state in the input configuration $C \bigcup C_{new\_sense}$. Each non-sink node x is labelled with the weight $\tilde{p}(x)$ as described above where we assume that state cns has higher probability than state cs. Function $\mathrm{connect\_SB}$ returns $C_{new\_com} = \{(10, cns), (11, cns)\}$. ■

5.4.4 Discussion

We remark that function E2P does not give false negatives. That is, it guarantees that any extensible input configuration C will be extended to a pathset by computing a suitable extension configuration C_{new}. This observation follows since:

1. Function $\mathrm{find\_SB}$ finds a set of sensing barrier paths $\{P_{sense}(X_i, \overline{X_i}) : i = 1, 2, \cdots\}$ in G_{sense} that satisfies condition (a) in Lemma 5.1 by

   (a) Excluding nodes in the input configuration C that are either in states $\{fail, cns\}$, and nodes that can not reach the sink in (G_{com}, C)

   (b) Using a shortest path algorithm to find each $P_{sense}(X_i, \overline{X_i})$ path

2. Function $\mathrm{connect\_SB}$ that adds new nodes to configuration $C \bigcup C_{new\_sense}$ to satisfy condition (b) in Lemma 5.1 ensures that each node in each $P_{sense}(X_i, \overline{X_i})$ path reaches the sink by a path of nodes in states $\{cs, cns\}$.

5.4.5 Running Time (of $\mathrm{E2P}$)

Denote by n the number of nodes in the given WSN. The running time of function $\mathrm{find\_SB}$ is determined by the loop in Steps 3 to 7. The loop iterates $|\mathbf{X}|$ times. Each iteration executes a most probable path algorithm (cf., Sec. 5.4.1). Since $|\mathbf{X}| \in O(n^2)$, function $\mathrm{find\_SB}$ requires $O(n^2 \cdot T_{SP}(n))$.

Similarly, the running time of function $\mathrm{connect\_SB}$ is determined by the loop in Steps 4 to 9. The loop iterates $|V_{SBP}|$ times. Each iteration executes a most probable path algorithm. Since $|V_{SBP}| \in O(n)$, function $\mathrm{connect\_SB}$ requires $O(n \cdot T_{SP}(n))$.

We conclude that function E2P requires $O(n^2 \cdot T_{SP}(n))$ time.

5.5 Optimal Extension to a Cutset

In this section, we examine the *optimal extension to a cutset* (E2C) problem that calls for computing a configuration C_{new} such that $C \bigcup C_{new}$ is a cutset, and the $\Pr(C_{new})$ is as high as possible. We recall that in Chapter 4, we have shown that the E2C problem for the BPDREL problem (where $G_{com} = G_{sense}$) can be solved exactly in polynomial time. In this section, we show a heuristic algorithm (function $\mathrm{E2C}$ below) that aims at computing an optimized solution whenever a solution exists. That is, the devised algorithm does not give false negatives.

5.5.1 Assumptions and Notations

We start by introducing a few assumptions and some needed definitions to present the algorithm. First, to simplify the presentation, we assume that for each node x, $x \neq sink$, $p_{com}(x), p_{sense}(x) \neq 0$ or 1. Indeed, a node x that has a boundary p_{com} or p_{sense} can be dealt with as special case as follows:

1. If ($p_{com}(x) = 0$ and $p_{sense}(x) = *$) then x is always in a failed state.
2. If ($p_{com}(x) = 1$ and $p_{sense}(x) = 0$) then x is always in the cns state.
3. If ($p_{com}(x) = 1$ and $p_{sense}(x) = 1$) then x is always in the cs state.
4. If ($p_{com}(x) = 1$ and $p_{sense}(x) \in (0, 1)$) then x can be assigned only to a state in $\{cs, cns\}$.

Our algorithm can distinguish each of the above cases and deal with it accordingly. We next introduce the following definitions.

1. $V_{cs,sink}$: A (possibly empty) subset of nodes where each node x is assigned the cs state, and x can reach the sink in G_{com} by a path of communication capable nodes (i.e., each internal node y on such a path is assigned a state $C(y) \in \{cs, cns\}$). An efficient algorithm to identify the set $V_{cs,sink}$ proceeds by modifying G_{com} by keeping all cs and cns nodes, and all nodes with $p_{com} = 1$, and deleting all free and failed nodes. We then check for cs nodes connected to the sink in the modified graph.

Rationale: Nodes in $V_{cs,sink}$ are **not** useful in computing a cutset extension C_{new}. Other cs nodes in the input configuration C can be disconnected from the sink by failing some free nodes; such cs nodes are considered by our algorithm in computing a solution C_{new}.

Example. In G_{com} of Fig. 5.5, assume that $C(2) = cs$ (note: the assumption is contrary to the figure where node 2 appears as a free node). Under this assumption, $V_{cs,sink} = \{2, 3, 4\}$. ■

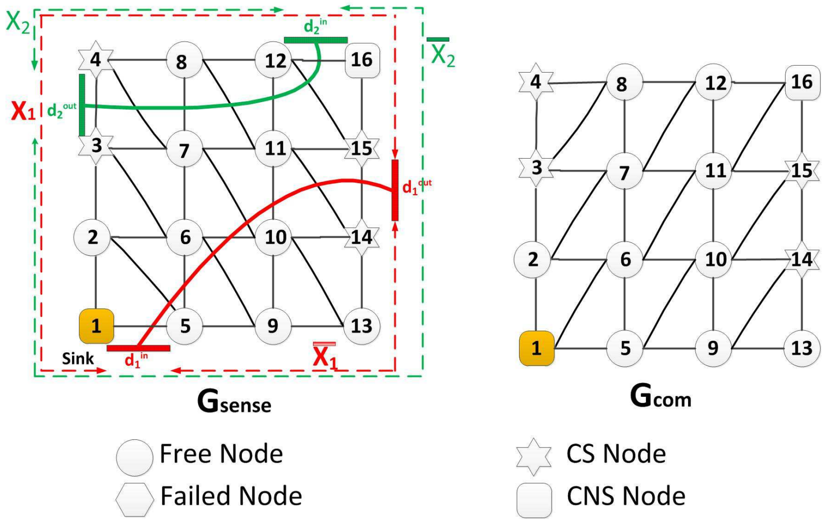


Figure 5.5: An example for function E2C

2. $P_{sense}(X, \overline{X})$: For a given pair of protection intervals $(X, \overline{X}) \in \mathbf{X}$, $P_{sense}(X, \overline{X})$ denotes any possible path in G_{sense} that connects some node $x \in X$ to some node $\overline{x} \in \overline{X}$. We require that any node v on such a path to be in state $C(v) \in \{cs, free\}$. That is, $P_{sense}(X, \overline{X})$ can be made a sensing barrier by assigning its free nodes to the cs state.

3. $V_{sense\_obstruct}(X, \overline{X})$: For a given $(X, \overline{X}) \in \mathbf{X}$, $V_{sense\_obstruct}(X, \overline{X})$ is a subset of nodes in $V \setminus V_{cs,sink}$ that intersects any possible $P_{sense}(X, \overline{X})$ path in at least one node such that we can obtain an $(X, \overline{X})$-cutset by disallowing

each node in $V_{sense\_obstruct}(X, \overline{X})$ from sensing and/or communicating with the sink.

Note that the definition restricts the search for $V_{sense\_obstruct}(X, \overline{X})$ to non-sink nodes assigned to one of $\{cs, free\}$ states in the configuration C. This follows since the sink node can not be assigned a failed or a non sensing state. And, a failed or cns node does not appear in any possible $P_{sense}(X, \overline{X})$ path.

Example. Fig. 5.5 illustrates G_{sense} of a WSN where the shape of each node specifies its state in the input configuration $C = \{(3, cs), (4, cs), (14, cs), (15, cs)\}$. For the pair $(X_1, \overline{X_1})$, it is possible to have

$$V_{sense\_obstruct}(X_1, \overline{X_1}) = X_1 = \{(5, free), (9, free), (13, free), (14, cs)\}$$

since any $P_{sense}(X_1, \overline{X_1})$ must include at least one node in X_1. ■

4. $V_{com\_obstruct}(t)$: For a given node t in the cs state, $V_{com\_obstruct}(t)$ is a set of free nodes that can be turned into failed nodes so as to disallow any operating $(sink, t)$-path in G_{com}.

Example. Fig. 5.5 illustrates G_{com} of the WSN in the previous example. For node $t = 14$, it is possible to have $V_{com\_obstruct}(t) = \{2, 5, 6\}$ since failing each of the three nodes disconnects t from the sink. ■

5.5.2 Function E2C

Function E2C presents the structure of the overall algorithm. Step 1 initializes the extension C_{new} to a failed value of $(-1, \emptyset)$. Step 2 computes the set $V_{cs,sink}$ from the input (G_{com}, C), as explained in the definition.

The algorithm then invokes the loop in Step 3 where it examines each pair $(X, \overline{X}) \in \mathbf{X}$ of protection intervals. The loop aims at computing an optimized $(X, \overline{X})$-cutset, denoted $C_{new}(X, \overline{X})$, if one exists. Each iteration of the loop proceeds in two phases. Step 5 implements **phase 1** where we aim at computing an optimized set $V_{sense\_obstruct}(X, \overline{X})$, as explained in Section 5.5.3.

```
Algorithm 8: Function E2C($G_{sense}, G_{com}, p, \mathbf{X}, C$)
Input: An instance $(G_{sense}, G_{com}, p, \mathbf{X}, C)$ of the E2C problem
Output: If the $E2C$ has a solution, return a configuration $C_{new}$ containing
        additional nodes used to form a cutset. Else, return an error value
        $(-1, \emptyset)$.
1 $C_{new} = (-1, \emptyset)$
2 Compute the set $V_{cs,sink}$
3 foreach (pair $(X, \overline{X}) \in \mathbf{X}$) do
4     set $C_{new}(X, \overline{X}) = \emptyset$
5     construct a flow network $(G'_{sense}, s, t, cap)$, as explained in Sec. 5.5.3,
        and find the maximum flow amount $f_{max}$
          1. if $(f_{max} == 0)$ { $C_{new}(X, \overline{X}) = \emptyset$; break }
          2. else if $(f_{max} \geq maxcap)$ { $C_{new}(X, \overline{X}) = (-1, \emptyset)$; continue }
          3. else set $V_{sense\_obstruct}(X, \overline{X})$ = nodes in a minimum $(s, t)$-cut in
            the flow network
6     foreach (cs node $t \in V_{sense\_obstruct}(X, \overline{X})$) do
7         construct a flow network $(G'_{com}, sink, t, cap)$, as explained in
            Sec. 5.5.4, and find the maximum flow amount $f_{max}$
              1. if $(f_{max} == 0)$ continue
              2. else if $(f_{max} \geq maxcap)$ { this case can not arise }
              3. else set $V_{com\_obstruct}(t)$ = nodes in a minimum $(s, t)$-cut in the
                flow network;
              set $C_{new}(X, \overline{X})$ +=$\{(x, fail) : x \in V_{com\_obstruct}(t)\}$
      end
8     foreach ($free$ node $x \in V_{sense\_obstruct}(X, \overline{X})$) do
              1. set $G'_{com} = G_{com}$ with all failed nodes in $C \bigcup C_{new}(X, \overline{X})$
                deleted
              2. if ($x$ can not reach the sink in $G'_{com}$) continue
              3. else set $C_{new}(X, \overline{X})$ += the best of $(x, fail)$ and $(x, cns)$
      end
  end
9 set $C_{new}$ = the best computed $C_{new}(X, \overline{X})$; return $(+1, C_{new})$
```

Steps 6, 7, and 8 implement **phase 2** that aim at making each node in $V_{sense\_obstruct}$ $(X, \overline{X})$ either failed, cns, or disconnected from the sink in G_{com}, as explained in Section 5.5.4. Finally, Step 9 sets C_{new} to the best computed configuration $C_{new}(X, \overline{X})$, and returns successfully.

5.5.3 Computing $V_{sense\_obstruct}(X, \overline{X})$

Step 5 of function E2C computes a valid set $V_{sense\_obstruct}(X, \overline{X})$ (if one exists). Recall that, by definition, any valid solution should not include the sink node, a failed or cns node, or any node in $V_{cs,sink}$.

Our approach transforms this problem to a maximum flow problem on an instance denoted (G'_{sense}, s, t, cap). Here, G'_{sense} is an undirected graph with two distinguished terminal nodes, denoted s and t, and each node x, $x \neq s, t$, has a real capacity, denoted $cap(x)$. A solver for the maximum (s, t)-flow problem also returns an (s, t)-cut.

We assume that each node x has $p_{com}(x), p_{sense}(x) \neq 0$ or 1. In general, the transformation assigns low capacities to nodes that we want to encourage their inclusion in the computed minimum (s, t)-flow-cut (e.g., free nodes with high failure probability). The transformation works as follows.

1. The graph G'_{sense} is constructed from G_{sense} by first deleting all nodes assigned either the failed or cns state in the input configuration C. We next add two new nodes s and t, and make s (respectively, t) adjacent to each node in the interval X (respectively, $\overline{X}$).

2. For each node x, $x \neq s, t$, in G'_{sense} we assign the following capacities based on the state $C(x)$ assigned to node x:

   (a) If $x \in V_{cs,sink}$, set $cap(x) = maxcap$ where $maxcap$ is a value that exceeds the sum of capacities of nodes not in this class

   (b) Else if $C(x) = free$ set $cap(x) = -\log \max(p_{fail}(x), p_{cns}(x))$

   (c) Else if $C(x) = cs$ then it is desirable to set $cap(x)$ based on the best set of free nodes that can be assigned the failed state so as to disconnect x from the sink in G_{com} when this action is needed in the loop of Step 6. However, such information is not available in phase 1. So, we choose to set $cap(x)$ to some acceptable and simple to calculate value. In particular, we calculate the average (over all free nodes in C) failure probability, denoted $\tilde{p}_{fail}$, and set $cap(x) = \tilde{p}_{fail}$.

A maximum (s,t)-flow solution, denoted f_{max}, corresponds to one of the following cases:

1. If $f_{max} = 0$, the failed and cns nodes in the input configuration C forbids the existence of any $P_{sense}(X, \overline{X})$. Thus, C is already a cutset. Step 5.1 sets $C_{new}(X, \overline{X}) = \emptyset$ and breaks from the main loop of Step 3. Step 9 then returns successfully with $C_{new} = \emptyset$.

2. If $f_{max} \geq maxcap$, any possible $P_{sense}(X, \overline{X})$ is composed of nodes in $V_{cs,sink}$. Thus, C is already an $(X, \overline{X})$-pathset. Step 5.2 sets $C_{new}(X, \overline{X})$ to an error value, and continues the main loop in Step 3 by considering the next pair $(X, \overline{X})$ of protection intervals.

3. If $0 < f_{max} < maxcap$, the corresponding (minimum capacity) (s,t)-cut is a valid $V_{sense\_obstruct}(X, \overline{X})$. Each node in such a set has a state in $\{cs, free\}$ with no node in $V_{cs,sink}$ (and no node is the sink node).

5.5.4 Computing $V_{com\_obstruct}(t)$

The loop in Step 6 iterates over all cs nodes in $V_{sense\_obstruct}(X, \overline{X})$. In each iteration, Step 7 computes a valid set $V_{com\_obstruct}(t)$ (if one exists) for a given target cs node t. We denote the active configuration at any possible iteration of the loop by $C_{iter} = C \bigcup C_{new}(X, \overline{X})$ (where $C_{new}(X, \overline{X})$ can potentially grow in each iteration).

We recall that, by definition, $V_{com\_obstruct}(t)$ is a set of $free$ nodes in the current configuration that can be assigned the failed state so as to disconnect t from the sink in G_{com}.

We transform the problem to a network flow problem. We denote the constructed instance by $(G'_{com}, s = sink, t, cap)$ where G'_{com} is an undirected graph, and each node x, $x \neq s, t$, has a real capacity, denoted $cap(x)$.

Similar to the previous section, we assume that each node x has $p_{com}(x), p_{sense}(x)$ $\neq 0$ or 1. Also, the transformation assigns low capacities to nodes that we want to encourage their inclusion in the computed minimum (s,t)-flow-cut (e.g., free nodes with high failure probability). The transformation works as follows.

1. G'_{com} is constructed from G_{com} by removing failed nodes in the current configuration C_{iter}.

2. For each node x, $x \neq s, t$, in G'_{com} we assign the following capacities:

   (a) If $C_{iter}(x) = free$, set $cap(x) = -\log p_{fail}(x)$

   (b) If $C_{iter}(x) \in \{cs, cns\}$, set $cap(x) = maxcap$ where $maxcap$ is a value that exceeds the sum of capacities of nodes not in this class

A maximum (s,t)-flow solution f_{max} corresponds to one of the following cases:

1. If $f_{max} = 0$, G'_{com} is disconnected, and no additional nodes need to be failed to disconnect t from the sink. Step 7.1 continues to the next iteration without augmenting $C_{new}(X, \overline{X})$ with additional nodes.

2. The case where $f_{max} \geq maxcap$ is not possible since by the selection of $V_{sense\_obstruct}(X, \overline{X})$, each included cs node can be disconnected from the sink in G_{com}.

3. If $0 < f_{max} < maxcap$ then we set $V_{com\_obstruct}(t)$ to all nodes in a minimum capacity (s,t)-cut of the flow network. All such nodes are free in the current configuration.

5.5.5 Correctness

In this section, we show that function E2C computes a valid extension C_{new} when one exists.

Lemma 5.3 *Function* E2C *computes a valid extension C_{new} if and only if the instance $(G_{sense}, G_{com}, p, \mathbf{X}, C)$ admits an extension $C \bigcup C_{new}$ to a cutset.*

Proof.
(⇒) Assume that function E2C has successfully computed an extension C_{new} for some pair $(X, \overline{X}) \in \mathbf{X}$ of protection intervals. We show that $C \bigcup C_{new}$ is an $(X, \overline{X})$-cutset. Thus, $C \bigcup C_{new}$ is a cutset.

To see that $C \bigcup C_{new}$ is an $(X, \overline{X})$-cutset, we note the following for the set $V_{sense\_obstruct}(X, \overline{X})$

1. By definition, any possible $P_{sense}(X, \overline{X})$ path in the input instance contains at least one cs or $free$ node in $V_{sense\_obstruct}(X, \overline{X})$.
2. Step 7 fails sufficient number of extra nodes to forbid any cs node in $V_{sense\_obstruct}$ $(X, \overline{X})$ from reaching the sink.
3. Step 8 assigns each free node in $V_{sense\_obstruct}(X, \overline{X})$ to either the cns, or the $fail$, state.

Thus, C_{new} is a valid extension as required.

($\Leftarrow$) Assume that $(G_{sense}, G_{com}, p, \mathbf{X}, C)$ admits an extension to a cutset $C \bigcup C_{new}$, we show that function E2C returns a solution. We distinguish the following cases.

1. **Case $C_{new} = \emptyset$ (i.e., C is already a cutset):** Step 5 constructs a network flow graph (G'_{sense}, s, t, cap), and we deal with the following cases:
   (a) **Case:** s does not reach t in G'_{sense} by a path of nodes in $\{cs, free\}$: Step 5.1 recognizes that S is already a cutset.
   (b) **Case:** G'_{sense} has an (s,t)-path composed of cs nodes, where no such node can reach the sink under the configuration C. The loop in Step 6 keeps $C_{new}(X, \overline{X}) = \emptyset$, and function E2C returns successfully with $C_{new} = \emptyset$, as required.
2. **Case $C_{new} \neq \emptyset$:** Any possible $P_{sense}(X, \overline{X})$ that can be turned into a sensing barrier (under configuration C) has at least one node x in $C \bigcup C_{new}$ where either
   (a) $(x, cs) \in C$ and C_{new} fails some nodes so as to disconnect x from the sink in $C \bigcup C_{new}$, or
   (b) $(x, fail)$ or $(x, cns) \in C_{new}$.

Denote by $V'_{sense\_obstruct}$ all such x nodes. Note that removing $V'_{sense\_obstruct}$ from the flow graph G'_{sense} disconnects s from t. So, $V'_{sense\_obstruct}$ is an (s,t)-cut in G'_{sense}. Thus, Step 5 succeeds in computing such a network flow cut. Steps 7 and 8 do not fail (all state assignments are feasible). We conclude that function E2C returns a valid extension C_{new}, as required. ■

5.6 Numerical Results

In this section, we investigate the performance of our approach for computing LBs and UBs for the DIR-BPDREL problem with 3-state nodes. We report on experiments done on WSNs formed by $W \times L$ grid nodes with W rows at (x,y)-coordinates $y = 0, 1, \cdots, W-1$, and L columns at coordinates $x = 0, 1, \cdots, L-1$. The sink node is located at the bottom left corner of the grid at coordinates $(0,0)$. The communication (or sensing) direction of a node is specified by the parameters $(r, \theta_{mid} \pm \alpha)$, as described in Section 5.1.1.

Note that the methodologies devised work for any pair of arbitrary G_{com} and G_{sense} graphs. In the graphs used in this section, nodes are arranged in a grid layout, however, some experiments vary the directionality parameters, and consequently the transmission and/or sensing radii are changed creating graphs that are less regular and much denser than regular grids.

Each WSN has one pair of entry-exit sides with a corresponding protection interval $(X, \overline{X})$, where X is the set of top row perimeter grid nodes, and $\overline{X}$ is the set of remaining perimeter nodes. By default, the algorithm executes 10,000 iterations to obtain a bound.

5.6.1 Exact reliability results

Here, we explore the range of grid sizes for which the algorithms compute $Rel(G_{com}, G_{sense})$ exactly in time less than 0.5 hour. In the experiments, $DIR_{com} = (1.7, 135° \pm 135°)$, and $DIR_{sense} = (0.5, 225° \pm 135°)$ (cf., the examples in Section 5.1.1). The LB (or UB) algorithm produces an exact result if it processes all of the generated configurations.

Table 5.1 presents the number of configurations generated by the LB and the UB algorithms, respectively. As can be seen, compared to the total number of states, the algorithms compute exact results using a significantly smaller number of configurations. Such reductions are achieved by avoiding the generation of configurations that can not be extended to either a pathset (for the LB algorithm), or a cutset (for the UB algorithm).

Table 5.1: Exact computations of the $\mathrm{DIR\text{-}BPDREL}$ problem

$W{\times}L$	**Network states**	**Generated configurations (LB)**	**Generated configurations (UB)**
2×2	3^3	17	17
2×3	3^5	43	21
3×3	3^8	395	167
3×4	3^{11}	2503	849
4×4	3^{15}	58199	21531
4×5	3^{19}	703411	384997

5.6.2 Gaps between LBs and UBs

Fig. 5.6 illustrates the gap between the obtained LBs and UBs for networks composed of $W{\times}W$ grid nodes, where W (on the x-axis) varies in the range $[2, 9]$. We set $p_{com} = p_{sense} = 0.8$ for each node. G_{com} and G_{sense} correspond to using $DIR_{com} = (1.7, 135° \pm 135°)$, and $DIR_{sense} = (0.5, 225° \pm 135°)$, respectively. The upper and lower curves in Fig. 5.6 correspond to bounds obtained using 10,000 iterations. The middle LB curve is obtained using 500,000 iterations (with a corresponding increase in the number of the stored configurations.)

The gap between the bounds given by the upper and lower curves increases as the network size increases. This follows since increasing the number of nodes causes a rapid increase in the number of pathsets and cutsets. So, 10,000 iterations fail short in covering a reasonable number of configurations. In addition, the size of each pathset or cutset increases, and hence the contribution of each such a configuration to the final answer decreases as well.

Our past experience, however, indicates that the LB curve tends to be more accurate than the UB curve. For example, by increasing the number of iterations

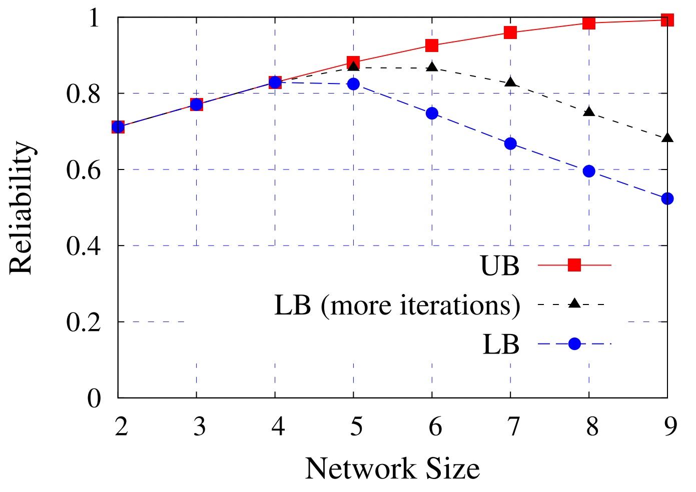


Figure 5.6: Effect of varying network size on LBs and UBs for the DIR-BPDREL problem

from 10,000 to 500,000, one obtains the middle curve in Fig. 5.6; this middle curve has a shape similar to the lower curve (with improved LB values). For networks with $W \geq 5$, each point of the middle curve requires about 7 hours of running time.

5.6.3 Effect of node communication and sensing probabilities

In this part, we explore the effect of varying node's p_{com} and p_{sense} on $Rel(G_{com}, G_{sense})$. We present numerical results of a WSN formed by 6x6 grid nodes, as a representative of other obtained results.

Fig. 5.7 illustrates the results. The dashed curve corresponds to setting $p_{com} = 0.6$, and varying $p_{sense} \in [0, 1]$ for all nodes. Similarly, the solid curve corresponds to setting $p_{sense} = 0.6$, and varying $p_{com} \in [0, 1]$ for all nodes.

The results show that the dashed curve has better performance than the solid curve when the x-axis values vary in the range $[0, 0.6]$. In this region, both curves result in nodes with the same p_{cs} $(= p_{com} \cdot p_{sense})$ probabilities. However, they differ widely in the produced p_{cns} $(= p_{com} \cdot (1 - p_{sense}))$ probabilities.

Specifically, the dashed curve results in p_{cns} varying in the range $[0.6 \times (1.0 - 0.6), 0.6 \times (1.0 - 0.0)]$ whereas the solid curve results in p_{cns} values varying in the range $[0.0 \times (1.0 - 0.6), 0.6 \times (1.0 - 0.6)]$. Thus, the dashed curve results in better

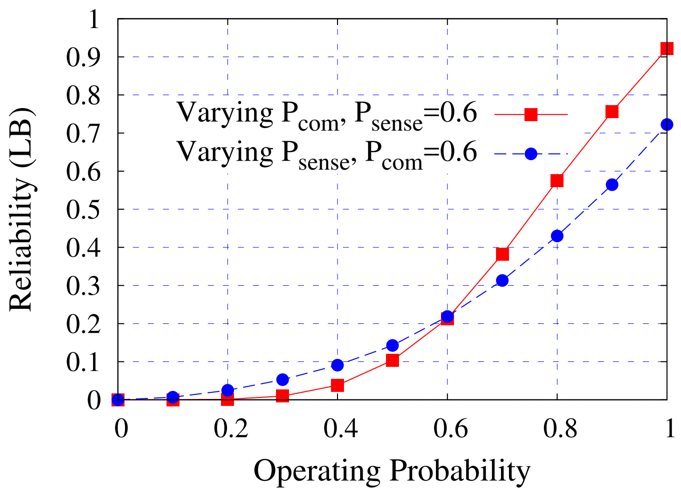


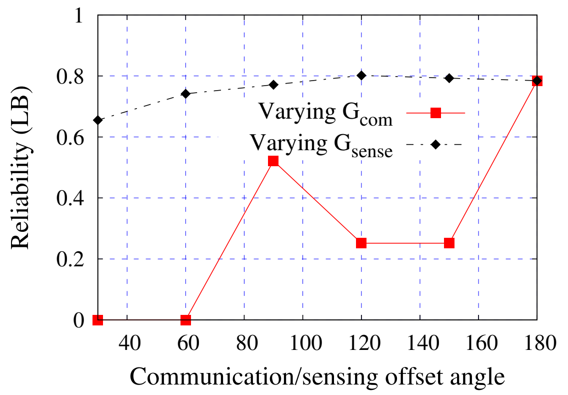
Figure 5.7: Effect of varying p_{com} and p_{sense} on $Rel(G_{com}, G_{sense})$ of the DIR-BPDREL problem

p_{cns} values.

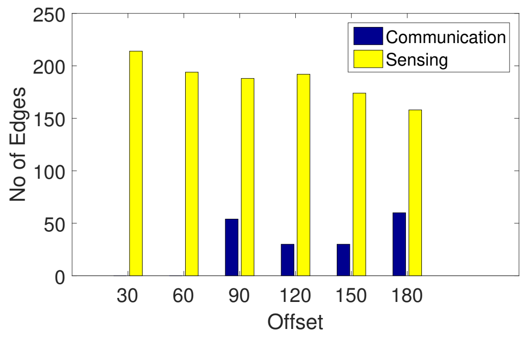
The p_{cns} probabilities are used in selecting relay nodes needed to construct pathsets. This explains the better performance of the dashed curve when the x-axis values vary in the range $[0, 0.6]$. Following a similar reasoning, one can explain the behaviour of the curves when the x-axis values vary in the range $[0.6, 1.0]$.

5.6.4 Effect of directional parameters

Directional communication and sensing devices can achieve spatial selectivity without significant changes in the consumed power. In this part, we explore the effect of changing DIR_{com} and DIR_{sense} on $Rel(G_{com}, G_{sense})$. We present numerical results of a WSN formed by 6x6 grid nodes. We set $p_{com} = p_{sense} = 0.8$ for all nodes. DIR_{com} and DIR_{sense} are specified by $(r = \frac{360}{2 \cdot \alpha}, 180° \pm \alpha)$, where the offset angle $\alpha \in \{30°, 60°,90°, 120°, 150°, 180°\}$. Fig. 5.8a illustrates the obtained results. The solid curve corresponds to setting $\alpha = 180°$ to obtain G_{sense}, and varying $\alpha \in \{30°, 60°, \cdots, 180°\}$ to experiment with different G_{com} graphs. In the histogram of Fig. 5.8b, columns with dark colour give the number of links in the constructed G_{com} graphs (in the range 0 to 60 links). We note that the performance given by the solid curve of Fig. 5.8a has a strong correlation to the number of



(a) Effect of varying the offset angle $\alpha \in \{30°, 60°, \cdots, 180°\}$ of DIR_{com} and DIR_{sense} on $Rel(G_{com}, G_{sense})$



(b) Number of links in different G_{com} and G_{sense} graphs obtained by varying the offset angle α

Figure 5.8: Effect of directional parameters on the obtained reliability of the DIR-BPDREL problem

links in G_{com}; that is, communication graphs with fewer edges give lower reliability values.

The dashed curve corresponds to setting $\alpha = 180°$ to obtain G_{com}, and varying $\alpha \in \{30°, 60°, \cdots, 180°\}$ to experiment with different G_{sense} graphs. In the histogram of Fig. 5.8b, columns with light colour give the number of links in the constructed G_{sense} graphs (in the range 158 to 214 links). In general, the obtained G_{sense} graphs have considerable number of links (≥ 158) that support the construction of many sensing barrier paths $P_{sense}(X, \overline{X})$. Here, the obtained LBs exceed 0.60. We note, however, when the offset angle α is small (e.g., $\alpha = 30°$) the sensing links are confined to a narrow region around each node compared to networks where α is large (e.g., the omnidirectional case where $\alpha = 180°$). Thus, the distribution of links in G_{sense} graphs obtained using large offset angles allows the construction of more sensing barrier paths. This explains the observed trend where large offset angles in G_{sense} result in better reliability values.

5.7 Concluding Remarks

In this chapter, we develop an approach for analyzing networks with directional sensor nodes deployed for an application where a WSN guards against unauthorized area traversal. Our model separates the network's sensing graph from its communication graph as a flexible way to model separate directional sensing and communication functions. In addition, the model separates events of communication failure from sensing failure within any single node. Under the above general model, we devise algorithms to assess the dependability of the network in such a critical application. The obtained results give lower and upper bounds on an underlying reliability measure.

Chapter 6

Breach Path to Target Area Detection Reliability in WSNs

In previous chapters, we have considered unauthorized traversal across a WSN area where the intruder path starts and ends outside the guarded area. In contrast, in this chapter, we consider intruder paths from outside the WSN area to a distinguished internal area of interest. The corresponding reliability problem is called the breach path to target area detection reliability ($\mathrm{BPTA\text{-}REL}$) problem. We adopt the framework used in the previous chapters as a main tool for computing lower and upper bounds on the current problem. Our main contributions lie in designing suitable algorithms for handling the extension to a pathset (E2P), and the extension to a cutset (E2C) problems for the current reliability problem. Some of the results in this chapter appear in [18]

6.1 Problem Formulation

6.1.1 Area Monitoring Model

We deal with a WSN modelled during a time period of interest as an undirected graph $G = (V \cup \{s\}, E)$ where V denotes the set of sensor nodes with a distinguished command and control sink node s, and E denotes a set of bidirectional links. Similar to the model used in Chapter 4, a link (x, y) is useful to us if it can sense an object crossing the line segment (x, y) from any point on the segment. Such link can guarantee detection of any crossing along the segment. So, we assume that all links in E satisfy this property.

We further assume that the WSN is deployed to protect a geographical area, denoted A_{wsn}. The area has a perimeter defined by a polygon that has a sensor node at each corner. The sides of the polygon does not necessarily correspond to communication or sensing links in G. Rather, the sides are used only to define the entry sides the intruder may use.

Breach paths are defined using two parameters: a set D of perimeter polygon sides, and an area A_{target} of interest. D represents a set of potential entry sides that an intruder can use. A_{target} lies within A_{wsn} and is defined by a polygon that has a sensor node at each corner. A *D-attack* is a crossing of the network from any side in D to A_{target} along any possible path across A_{wsn}. The WSN provides the required protection if it can detect and report to the sink any D-attack. Sensor nodes that may lie inside A_{target} are not useful for successful network operation, and hence we may assume that no such sensor node exists.

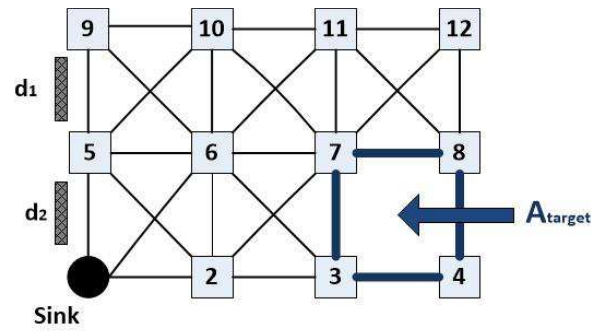


Figure 6.1: An instance of the BPTA-REL problem

Example. Fig. 6.1 illustrates a WSN where $|V| = 11$ nodes, $D = \{d_1, d_2\}$, and A_{target} is defined by the polygon $(3, 7, 8, 4)$. ■

6.1.2 Reliability Model

Similar to Chapter 4, we associate with each node x, an operation (failure) probability $p(x)$ (respectively, $q(x) = 1 - p(x)$). Nodes are assumed to operate independently of each other. The sink is typically well maintained and protected, so $p(s) = 1$.

A failure event of one, or more nodes, leaves the network with a subset S of operating nodes where the remaining nodes $V \setminus S$ are failed. We refer to such set S as a network *state*. Thus a network on n nodes has 2^n different states, where each state S arises with probability $\Pr(S) = \prod_{x \in S} p(x) \prod_{x \notin S} q(x)$. In our present context, a state S is *operating* if it can detect and report to the sink any D-attack. Else, S is *failed*. In Fig. 6.1, $D = \{d_1, d_2\}$ and the state $S = \{5, 9\}$ (with other non-sink nodes failed) is an operating state. The BPTA-REL problem can then be defined as follows.

Definition (the BPTA-REL **problem).** Let $G = (V \cup \{s\}, E)$ be a WSN where each node x, $x \neq s$, has an operating probability $p(x)$, and each point on a line segment $(x, y) \in E$ can be detected by either x and/or y. In addition, let D be a set of entry sides on the perimeter of G, and A_{target} be an area of interest within the network. Find the probability $Rel(G, p, D, A_{target})$ that G is in a state that ensures that any D-attack is detected. ■

6.1.3 Complexity Analysis

Theorem 6.1 *The* BPTA-REL *problem is #P-hard.*

Proof. The proof is similar to the proof of Theorem 4.1. Here, we reduce in polynomial time a given instance (G, s, t) of the $2REL$ problem on grid networks to an instance (G', p, D, A_{target}) of the BPTA-REL problem such that $Rel(G, s, t) = Rel(G', p, D, A_{target})$. Figure 6.2 illustrates the reduction where nodes s and t are assumed to be two non-adjacent nodes on the perimeter of a partial grid network G. In Fig. 6.2, we have the following:

1. The probabilistic graph G' of the BPTA-REL problem is constructed from the graph G by adding two new nodes a and b as shown in the figure where $p(a) = p(b) = 1$. The figure shows A_{target} and d_{in}.

2. Node t is taken as the sink node of the BPTA-REL instance.

3. Any node $x \neq t, a,$ or b in G' has the same operating probability $p(x)$ as in G.

4. An operating state of the BPTA-REL problem on the graph G' is a state where an intruder crossing G' from side d_{in} to A_{target} can be detected and reported. Note that, any such intruder path has to enter the network from d_{in}, and enter A_{target} by crossing the link (s, c). Intrusion paths of the problem can not leave the network and then enter A_{target} from any of the 3 sides (s, a), (a, b), or (b, c).

Thus, there is a one-to-one correspondence between operating states of the BPTA-REL problem in G' and operating states of the $2REL$ problem in G, as required. ■

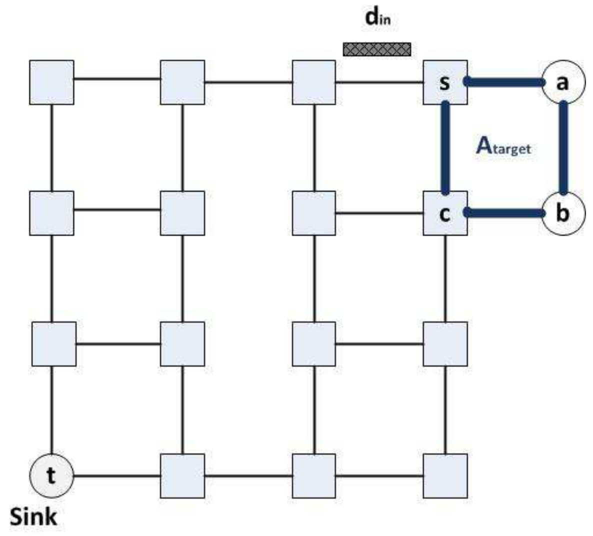


Figure 6.2: The graph G' used in Theorem 6.1

6.2 Overview of the Main Method

Our main method of deriving lower and upper bounds (LBs and UBs) on the exact solution utilizes the concepts of network *configurations*, *pathsets*, and *cutsets* introduced below.

Network configurations. A configuration C of a network assigns a state from the set $\{op, fail\}$ to each node in some subset of nodes. We use $V(C) \subseteq V$ to denote such subset. Non-sink nodes that are not assigned a state in C are *free* nodes. In Fig. 6.1, $C = \{(5, op), (9, op)\}$ is a possible network configuration that has 9 free nodes. For convenience, we use C_{op}, C_{fail}, and C_{free} to denote the operating, failed, and free nodes in C. The probability that such configuration arises is $\Pr(C) = \prod_{x \in C_{op}} p(x) \cdot \prod_{x \in C_{fail}} q(x)$.

Pathsets. A $\mathrm{BPTA\text{-}REL}$ pathset is an operating configuration C. That is, C guarantees to detect and report to the sink any D-attack. In Fig. 6.1, $C = \{(5, op), (9, op)\}$ is a pathset.

Cutsets. A $\mathrm{BPTA\text{-}REL}$ cutset is a configuration C that can not be extended to a pathset by operating all possible free nodes. In Fig. 6.1, $C = \{(5, fail), (6, fail), (7, fail)\}$ is a cutset.

Our method relies on computing and using pairwise s-disjoint configurations as follows. Suppose that $\{P_1, P_2, \cdots, P_r\}$ is a set of pairwise s-disjoint pathsets then $Rel(G) \geq \sum_{i=1}^{r} \Pr(P_i)$ (the RHS is the computed LB). Likewise, suppose that $\{U_1, U_2, \cdots, U_r\}$ is a set of pairwise s-disjoint cutsets then $Rel(G) \leq 1 - \sum_{i=1}^{r} \Pr(U_i)$ (the RHS is the computed UB).

Several algorithms can be used to systematically generate a maximal set of pairwise pathsets (or cutsets). If the RHS in the first (second) relation is a maximal set of pathsets (respectively, cutsets) then the relation holds as an equality, and we obtain an exact solution on the problem. Our work in chapter 3 presents one such generation algorithm developed for networks where each sensor node can be in any one of three possible states. In this chapter, our main method utilizes a similar algorithm adapted to networks with 2-state (operating/fail) nodes. One important ingredient in the effective use of any such algorithm, however, is the ability to extend (if possible) any given configuration C to a pathset (or a cutset) that has a high occurrence probability. The next sections deal with this aspect for the $\mathrm{BPTA\text{-}REL}$ problem.

6.3 Planar Duality Approach

Our approach for extending a given configuration to a pathset (or cutset) relies on using certain planar graph duality relations between two graphs, denoted H and H^*, derived from the WSN G (see, e.g., [10] for background on planar graphs). In this section, we introduce the needed concepts.

6.3.1 Graphs H and H^*

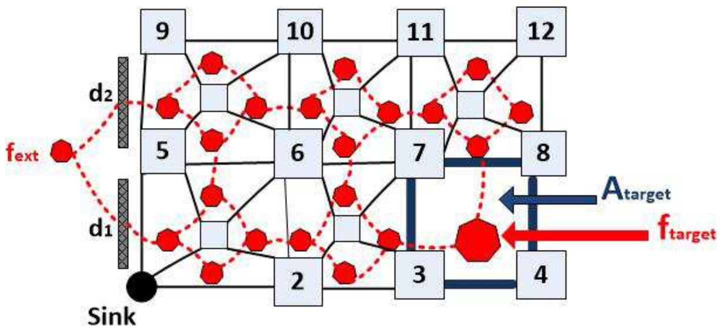


Figure 6.3: An instance of the $\mathrm{BPTA\text{-}REL}$ problem

The planar graph $H = (V_H, E_H)$ is constructed as follows. Consider an embedding of G in the plane where each node is placed according to its (x, y)-coordinates. The perimeter of G in this embedding forms a polygon surrounding the area A_{wsn}. Delete from the graph all links and sensor nodes that may lie within A_{target}. The remaining links of G may intersect each other. Take each intersection point (between two, or more links) as a new node in H. Thus, each link e in H is either a whole link in G, or part of a link in G. Conversely, each link e in G corresponds to a path of one, or more, links in H. The area of interest A_{target} appears as a face of H, denoted f_{target}. We also use f_{ext} to denote the exterior unbounded face of H.

To construct the planar graph H^*, we first take the planar dual of H. Next, for every perimeter side d of H that is not an entry side in D, we delete the dual link d^* from H^*.

Example. In Fig. 6.3, the graph H derived from the network in Fig. 6.1 has square nodes and solid links. The graph H^* has octagonal nodes and dashed lines. ■

6.3.2 Link Correspondence Relations

Our devised method uses the graph H^* to compute important structures on the network G. The following relations are thus important.

The $G2H^*$ Relation. If (x, y) is a link in G then (x, y) corresponds to an (x, y)-path in H. Each link e in such path corresponds to a dual link e^* is H^*. We denote such set of links in H^* by $G2H^*(x, y)$. Thus, link (x, y) in G corresponds to one, or more, links $G2H^*(x, y)$ in H^*.

The H^*2G Relation. If e^* is a link in H^* then e^* has a dual link e in H. Link e corresponds to either a whole or part of a link (x, y) in G. We denote such link (x, y) by $H^*2G(e^*)$. Thus, link e^* corresponds to link $H^*2G(e^*)$ in G.

6.3.3 Pathset Characterization

Our algorithm for finding pathsets utilize the characterization in Theorem 6.2 below. To start, we need the following terminology.

- A (D, A_{target})-*barrier* in G is a set of links that intersects any intruder path from any possible entry side in D to A_{target}.
- A (f_{ext}, f_{target})-*cut* in H^* is a set of links that forms a cut separating node f_{ext} from node f_{target}.

Example. In Fig. 6.1, the set $E'_G = \{(2, 6), (6, 10)\}$ forms a (D, A_{target})-barrier in G. The corresponding 2 links in H^* form a (f_{ext}, f_{target})-cut in H^*. ■

The duality relation between H and H^* allows us to state the following property.

Theorem 6.2 *A set of links forms a (D, A_{target})-barrier in G iff their corresponding links in H^* form a (f_{ext}, f_{target})-cut.*

6.3.4 Cutset Characterization

Likewise, our algorithm for finding cutsets utilizes the characterization in Theorem 6.3. We need the following terminology.

- A (D, A_{target})-*breach passage* in G is a set of links whose failure allows an undetected intrusion path. (A link e in G fails if at least one of its two end nodes fail, or become disconnected from the sink.)
- A (f_{ext}, f_{target})-*path* in H^* is a set of links that form a path from node f_{ext} to node f_{target}.

Example. In Fig. 6.1, the set E'_G on 9 links formed by the 3 vertical links $(5, 9)$, $(6, 10)$, $(7, 11)$, the 5 diagonal links $(5, 10)$, $(6, 9)$, $(6, 11)$, $(7, 10)$, $(7, 12)$, and the horizontal link $(7, 8)$ forms a (D, A_{target})-breach passage in G. The corresponding links in H^* contain multiple (f_{ext}, f_{target})-paths. ■

Likewise, the duality relation between H and H^* allows us to state the following property.

Theorem 6.3 *A set of links form a (D, A_{target})-breach passage in G iff their corresponding links in H^* contain a (f_{ext}, f_{target})-path.*

6.4 Extension to a Pathset

The optimal *extension to a pathset* (E2P) problem is defined as follows. Given an instance of the BPTA-REL problem, and a configuration C, extend C if possible to a pathset $C \bigcup C_{new}$ so that $\Pr(C_{new})$ is as high as possible. Our contribution in this section is function E2P that computes an effective solution to the problem. The function is guaranteed to find a feasible solution if one exists.

Our approach uses the following ideas:

- We first recall that if $C \cup C_{new}$ is a pathset then, by definition, some links in $E_G(C \cup C_{new})$ form a (D, A_{target})-barrier in G.

- Our strategy in constructing C_{new} relies on computing an optimized barrier in G obtained by solving a maximum flow problem on H^*.
- To this end, the function assigns to each link e^* in H^* that is derived from a link (x, y) in G (i.e., $H^*2G(e^*) = (x, y)$), a capacity $\mathrm{cap}(x, y) \geq 0$. This step is detailed in Sec. 6.4.2.
- Our goal is to set $\mathrm{cap}(x, y)$ to a small value whenever x and y have a high probability of operating and reaching the sink in G. Assigning e^* a small capacity encourages its inclusion in a minimum capacity (f_{ext}, f_{target})-cut of graph H^*. Including e^* in the computed minimum cut causes the function to select nodes x and y for inclusion in the constructed barrier, and consequently in the computed C_{new}, as desired.

6.4.1 Function E2P

We now give an overview of the overall structure of function E2P. More details appear in Sec. 6.4.2.

Step 1 constructs graphs H and H^*. Step 2 assigns to each link e^* in H^* that is derived from a link $H^*2G(e^*) = (x, y)$ in G, a capacity $\mathrm{cap}(x, y) \geq 0$. Our method of computing such capacity (cf. Sec. 6.4.2) relies on computing for each node x a best path $P_{s,x}$ (i.e., a path with highest possible probability) for reaching the sink in G. We denote the probability that all free nodes on such path operate by $p_{conn}(x)$.

Step 3 solves an instance of the maximum flow problem on H^* to identify a minimum capacity (f_{ext}, f_{target})-cut, denoted E'_{H^*}.

Step 4 returns from the function in two special cases. If node f_{ext} is already disconnected from node f_{target} in H^* then the computed minimum cut is empty (i.e., $E'_{H^*} = \emptyset$). In this case, C is already a pathset that does not require any extension. On the other hand, if the computed cut contains a link e^* that corresponds to a failed link (x, y) in configuration C then C is already a cutset. This latter condition exists when $\mathrm{cap}(E'_{H^*}) \geq \mathrm{MAXCAP}$, where MAXCAP is a large capacity defined in Sec. 6.4.2.

Step 5 processes links of the minimum cut E'_{H*}, and identifies two subsets of

```
Algorithm 9: Function E2P($G, p, D, A_{target}, C, C_{new}$)
Input: Instance $(G, p, D, A_{target}, C)$ of the $E2P$ problem
Output: Return $+1$ and a solution $C_{new}$ if possible. Else, $C$ is a cutset, return
        $-1$.
1 Construct graphs $H$ and $H^*$.
2 Assign capacities:
      For each link $e^*$ in $H^*$ corresponding to a link $H^*2G(e^*) = (x, y)$ in $G$,
        assign a capacity $\mathrm{cap}(x, y) \geq 0$, as explained in Sec. 6.4.2.
3 Compute minimum cut:
      Find a minimum capacity $(f_{ext}, f_{target})$-cut in $H^*$. Denote the links of
        such minimum cut by $E'_{H^*}$.
4 if ($E'_{H^*} == \emptyset$) then
      set $C_{new} = \emptyset$; return $+1$
  else if ($\mathrm{cap}(E'_{H^*}) \geq \mathrm{MAXCAP}$) then
      return $-1$
  end
5 Build initial $V_{free}$ set:
      Process each link $e^*$ in the computed minimum cut $E'_{H^*}$ (cf. Sec. 6.4.2).
        This step gives two sets: $V_{op} \subseteq C_{op}$, and $V_{free} \subseteq C_{free}$.
6 Augment $V_{free}$:
      Compute a set of additional free nodes $V'_{free}$ that suffices to connect each
        node in $V_{op} \bigcup V_{free}$ to the sink using a path composed of nodes in
        $C_{op} \bigcup V_{free} \bigcup V'_{free}$. Add $V'_{free}$ to $V_{free}$.
7 Refine $V_{free}$:
      Remove from $V_{free}$ nodes that are not necessary to form a pathset.
8 Set $C_{new} = V_{free}$ after assigning them the operating state; return $+1$
```

nodes: a set of operating nodes $V_{op} \subseteq C_{op}$, and a set of free nodes $V_{free} \subseteq C_{free}$, as explained in Sec. 6.4.2.

Step 6 computes a set, denoted V'_{free}, containing possibly additional free nodes. With the help of nodes in this set, every node in $V_{op} \bigcup V_{free}$ can reach the sink by a path composed of nodes in $C_{op} \bigcup V_{free} \bigcup V'_{free}$. Initially, V'_{free} is empty. We subsequently add free nodes to the current set V'_{free} by iteratively processing each node $x \in V_{op} \bigcup V_{free}$. In more detail, if $P_{s,x}$ denotes the best path connecting the sink to x in G then we add to V'_{free} all possible free nodes of $P_{s,x}$ that are not in $V_{free} \bigcup V'_{free}$. To encourage the inclusion of free nodes with high operating probabilities in the computed set V'_{free}, we process the nodes in $V_{op} \bigcup V_{free}$ in decreasing order of their p_{conn} probabilities. Subsequently, Step 6 adds V'_{free} to V_{free}.

Following Step 6, it is possible that some nodes in V_{free} are unnecessary for forming a pathset. Step 7 removes such extra free nodes. This is done by iteratively deleting a free node from V_{free}, and testing whether the remaining nodes form a pathset. Thus, a superfluous node is removed from V_{free} before the next iteration. Finally, Step 8 returns the computed solution.

6.4.2 Link Capacity Assignment

As mentioned above, we assign to each link e^* in H^* that corresponds to a link $H^*2G(e^*) = (x, y)$ in G, a capacity $\mathrm{cap}(x, y) \geq 0$. Our goal is to set $\mathrm{cap}(x, y)$ so as to favour the inclusion of node x (and/or y) in the solution C_{new} if **(a)** x contributes to building a (D, A_{target})-barrier in G, and **(b)** x is a an operating or free node in C that has a high probability of reaching the sink node. We have synthesized and experimented with a number of capacity functions. We present below a function that has given us the best results.

We start by revising node operation probabilities according to the given input configuration C. Specifically, for each node $x \in C_{op}$ (or, $x \in C_{fail}$), we set $p(x) = 1$ (respectively, $p(x) = 0$). Next, we note that if we select a node x (operating or free) in constructing a barrier then there may be an additional cost of using this node incurred by establishing an operating path from the sink to x. To analyze such cost, we introduce the following notation.

- $\Pr(P_{s,x})$: the probability that node x operates and reaches the sink by a specified path $P_{s,x}$. Using the revised node operation probabilities, we have $\Pr(P_{s,x}) = \prod_{y \in V(P_{s,x})} p(y)$.
- $p_{conn}(x)$: the highest probability that node x operates and reaches the sink. More specifically, if $\mathbf{P}_{s,x}$ is the set of all possible (s, x)-paths whose internal nodes are either operating or free then

$$p_{conn}(x) = \max_{P_{s,x} \in \mathbf{P}_{s,x}} \Pr(P_{s,x}).$$

As can be seen, $p_{conn}(x)$ represents the best cost of operating node x, and linking it to the sink. If C_{new} is the computed solution then this cost appears as a multiplicative factor in $\Pr(C_{new})$. Thus, it is important to make node selection decisions based on using optimized costs. We note that the problem of computing optimal p_{conn} probabilities for free nodes is essentially a single-source shortest paths problem on the graph G. Thus, all such optimum paths and probabilities can be computed efficiently. Subsequently, our algorithm assigns to each link e^* a capacity:

$$\text{cap}(x, y) = -\log(p_{conn}(x) \cdot p_{conn}(y)).$$

We now exhaust all possible cases of links in H^*. In each case, we explain the suitability of the above function, and the steps taken to construct sets V_{op} and V_{free} mentioned in Step 5.

Case 1: $p_{conn}(x) \cdot p_{conn}(y) \neq 0$ or 1 (so, $\text{cap}(x, y) \neq \infty, 0$). In this case, each node (x or y) is either operating or free in C. Step 5 adds each such node to V_{op} (V_{free}) if the node is in C_{op} (respectively, C_{free}). We note that operating the link (x, y) in G requires operating and linking each of x and y to the sink. This aspect is considered in Step 6. We also note that the capacity function associates relatively small $\text{cap}(x, y)$ when $p_{conn}(x) \cdot p_{conn}(y)$ assumes relatively large value. Thus, encouraging the use of nodes with high joint operating and reaching the sink probabilities in the computed (D, A_{target})-barrier.

Case 2: $p_{conn}(x) \cdot p_{conn}(y) = 0$ (so, $\text{cap}(x, y) = \infty$). This case arises if x (and/or y) is assigned a failed state in C, or the node is not failed but it can not reach the sink through any set of operating/free nodes. Here, it is not possible to operate link (x, y) in G. We set $\text{cap}(x, y)$ to a large value, denoted MAXCAP, that is larger than the sum of all link capacities in Case 1. Step 4 detects the use of any such link in the computed minimum cut.

Case 3: $p_{conn}(x) \cdot p_{conn}(y) = 1$ (so, $\text{cap}(x, y) = 0$). This case arises if both x and y are assigned the operating state in C, and both nodes reach the sink by paths whose internal nodes are all operating. Here, link (x, y) is ready to be used in a barrier, and there is no need to include the corresponding link e^* in the computed minimum cut. The capacity function captures this aspect by setting $\mathrm{cap}(x, y) = 0$, so as to avoid using it. All such links are deleted from H^* prior to solving the maximum flow problem.

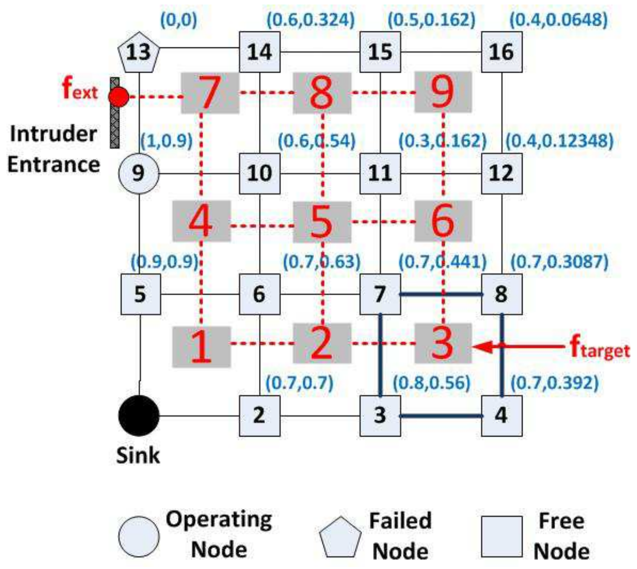


Figure 6.4: Pathset extension example for the BPTA-REL problem

Example. Fig. 6.4 illustrates an instance of the E2P problem where D has one entrance side, $A_{target} = f_3$, and $C = \{(9, op), (13, fail)\}$. All other non-sink nodes are free in C. In the top right of each node x, we show $(p(x), p_{conn}(x))$. The dual graph H^* has links represented by dashed lines, and nodes represented by colored squares. We note the following.

- Steps 1 to 3 compute the shown p_{conn} probabilities, and a minimum (f_{ext}, f_{target})-cut $E'_{H^*} = \{(f_4, f_7), (f_7, f_8)\}$.
- Step 5 sets $V_{op} = \{9\}$, and $V_{free} = \{10, 14\}$. This follows since link (f_4, f_7) in H^* corresponds to link $(x = 9, y = 10)$ in G (Case 1 applies). Similarly, link (f_7, f_8) in H^* corresponds to link $(x = 10, y = 14)$ in G (Case 1 applies too).
- Step 6 sets $V'_{free} = \{5\}$. This setting suffices to connect the sink to each node in $V_{op} \bigcup V_{free} = \{9, 10, 14\}$ by a path composed of nodes in $C_{op} \bigcup V_{free} \bigcup V'_{free} = \{9, 10, 14, 5\}$, as required. Subsequently, Step 6 extends V_{free} to $\{10, 14, 5\}$.

- Step 7 does not modify V_{free}. Thus, $C_{new} = \{(10, op), (14, op), (5, op)\}$

6.4.3 Correctness

Theorem 6.4 *Function* E2P *computes a feasible extension C_{new} if and only if such extension exists.*

Proof. We first show that if E2P returns $+1$ then $C \bigcup C_{new}$ is a BPTA-REL pathset. Here, Step 3 computes a minimum (f_{ext}, f_{target})-cut in H^* whose links are denoted E'_{H^*}. The set E'_{H^*} corresponds to a set E'_G of links in G through the H^*2G relation. Theorems 6.2 ensure that E'_G, with links joining the subset of nodes in C_{op} that can reach the sink in G, form a (D, A_{target})-barrier in G. Thus, if all nodes in E'_G operate and reach the sink then any intrusion from the set D of entrance sides to A_{target} can be detected and reported to the sink. Step 4 handles the case where no links need to be added to E'_{H^*} to form a cut (the case where $E'_{H*} = \emptyset$), and thus, configuration C is already a pathset. Otherwise, following Step 6, we know that operating all nodes in $V_{op} \bigcup V_{free}$ guarantees that all nodes in E'_G operate and reach the sink. Thus, assigning the operating state to all nodes in V_{free} results in a configuration that extends C to a pathset. Step 7 refines V_{free} without violating the above property, as required.

Next, we show that if configuration C admits an extension C_{new} so that $C \bigcup C_{new}$ is a pathset then function E2P returns $+1$ and a possible solution. To this end, denote by V_{conn} the subset of nodes in $C \bigcup C_{new}$ where each node operates and reaches the sink by other nodes in V_{conn}. By definition, the subgraph induced by V_{conn} in G contains a set of links, denoted E'_G, that forms a (D, A_{target})-barrier. Nodes in C_{new} appear in the input of function E2P as free nodes. So, the function sees nodes incident to E'_G as either free nodes, or operating nodes that may not be able to reach the sink using nodes in C_{op}. Note that E'_G has no link (x, y) that satisfies Case 2 (i.e., $\mathrm{cap}(x, y) = \mathrm{MAXCAP}$). The set E'_G corresponds to a set of links in H^*, denoted E'_{H^*}, by the $G2H^*$ relation. Theorem 6.2 ensure that E'_{H^*} is a (f_{ext}, f_{target})-cut in H^*. As noted above, no link in such cut is assigned the MAXCAP capacity. Deleting links from H^* (as done in Case 3) preserves the property that the resulting graph has a cut that does not use any link of capacity MAXCAP. The minimum cut computation in Step 3 is guaranteed to find such cut. Thus, $\mathrm{E2P}$ returns $+1$ and a feasible solution, as required. ■

6.5 Extension to a Cutset

Similar to the E2P problem, we define the optimal *extension to a cutset* (E2C) problem as follows. Given an instance of the $\mathrm{BPTA\text{-}REL}$ problem, and a configuration C, extend C if possible to a cutset $C \cup C_{new}$ so that the occurrence probability $\Pr(C_{new})$ is as high as possible. Thus, building C_{new} favours the inclusion of nodes with high failure probabilities.

Our contribution in this section is function $\mathrm{E2C}$ for solving the problem. The algorithm is not optimal but guarantees to find a feasible solution if one exists. Our design relies on the following insights:

- We first recall that if $C \cup C_{new}$ is a cutset then, by definition, some links in $E_G(C \cup C_{new})$ form a (D, A_{target})-breach passage.
- Our strategy in choosing C_{new} relies on computing such passage mainly by solving a shortest path problem on H^*.

In more detail, the function assigns to each link e^* in H^* that is derived from a link (x, y) in G (i.e., $H^*2G(e^*) = (x, y)$), a distance $dist(x, y) \geq 0$. Sec. 6.5.2 presents a particular distance function used for this propose. Roughly speaking, the utilized function gives a short distance whenever x (and/or y) is perceived to be highly useful in constructing a (D, A_{target})-breach passage in G. Such short distance assignment has the effect of giving link e^* a good chance of being selected in a shortest (f_{ext}, f_{target})-path in graph H^*. Choosing link e^* in such path results in function $\mathrm{E2C}$ selecting node x (and/or y) for inclusion in constructing the required breach passage, as desired.

6.5.1 Function $\mathrm{E2C}$

Step 1 constructs graphs H and H^*. Step 2 associates with each link e^* in H^* that is derived from a link $H^*2G(e^*) = (x, y)$ in G, a distance $dist(x, y) \geq 0$, as

```
Algorithm 10: Function E2C($G, p, D, A_{target}, C, C_{new}$)
Input: Instance $(G, p, D, A_{target}, C)$ of the $E2C$ problem
Output: Return $+1$ and a solution $C_{new}$ if possible. Else, $C$ is pathset, return
        $-1$.
1 Construct graphs $H$ and $H^*$.
2 Assign distances:
      For each link $e^*$ in $H^*$ corresponding to a link $H^*2G(e^*) = (x, y)$ in $G$,
        assign a distance $dist(x, y) \geq 0$, as explained in Sec. 6.5.2.
3 Compute shortest path:
      Find a shortest $(f_{ext}, f_{target})$-path in $H^*$. Denote the links of such
        shortest path by $E'_{H^*}$.
4 if ($\text{dist}(E'_{H*}) == 0$) then
  |   set $C_{new} = \emptyset$; return $+1$
  else if ($E'_{H^*} == \emptyset$) then
  |   return $-1$
  end
5 Build initial $V_{free}$ set:
      Process each link $e^*$ in the computed shortest path $E'_{H^*}$ (cf. Sec. 6.5.2).
        This step gives two sets: $V_{op} \subseteq C_{op}$, and $V_{free} \subseteq C_{free}$.
6 Augment $V_{free}$:
      Compute a set of additional free nodes $V'_{free}$ so that failing all nodes in
        $V_{free} \bigcup V'_{free}$ disconnect all nodes in $V_{op}$ from the sink. Add $V'_{free}$ to
        $V_{free}$.
7 Refine $V_{free}$:
      Remove from $V_{free}$ nodes that are not necessary to form a cutset.
8 Set $C_{new} = V_{free}$ after assigning them the failed state; return $+1$
```

explained in Sec. 6.5.2. Step 3 computes a shortest (f_{ext}, f_{target})-path in graph H^*. We denote the links of such path by E'_{H^*}.

Step 4 handles two special outcomes. If the computed path has zero length (i.e., $\text{dist}(E'_{H*}) = 0$) then configuration C is already a cutset. On the other hand, if no such path exists (i.e., $E'_{H^*} = \emptyset$) then C is already a pathset.

Step 5 inspects the computed path E'_{H^*}, and identifies two subsets of nodes: a set of operating nodes $V_{op} \subseteq C_{op}$, and a set of free nodes $V_{free} \subseteq C_{free}$, as explained in Sec. 6.5.2

Step 6 computes a set, denoted V'_{free}, of additional free nodes such that if all nodes in $C_{fail} \bigcup V_{free} \bigcup V'_{free}$ fail then no node in V_{op} can possibly reach the sink. Initially, V'_{free} is empty. We subsequently add free nodes to V'_{free} by iteratively disconnecting each node $x \in V_{op}$. To encourage the inclusion of free nodes with high failure probabilities in V'_{free}, we use the following observation: computing the best subset of free nodes whose failure disconnects x from the sink in G can be transformed to computing a minimum capacity cut of a network flow problem on graph G. In the transformation, capacities are assigned to nodes of G. Any possible additional free node computed in this step is added to V'_{free}. Subsequent iterations to disconnect the remaining nodes in V_{op} benefit from nodes currently in $V_{free} \bigcup V'_{free}$. Step 6 then adds V'_{free} to V_{free}.

Following Step 6, it is possible that some nodes in V_{free} are unnecessary for forming a cutset. Step 7 removes such extra free nodes. This is done by iteratively deleting a free node from V_{free}, and testing whether the remaining nodes form a cutset. Thus, a superfluous node is removed from the set of free nodes before the next iteration. Finally, Step 8 returns the computed solution.

6.5.2 Link Distance Assignment

Throughout this section, we assume that e^* is a link in H^* that corresponds to a link $H^*2G(e^*) = (x, y)$ in G. Function E2C assigns to e^* a distance $dist(x, y) \geq 0$. Our goal is to set such distance so as to favour the inclusion of node x (and/or y) in the solution C_{new} when

(a) x contributes to building a (D, A_{target})-breach passage in G, and

(b) x is a free node in C with relatively high failure probability, or x is an operating node in C that can be disconnected from the sink by failing some free nodes of relatively high failure probability.

To serve this purpose, we start by revising node failure probabilities to reflect the assignments made in the input configuration C. Namely, we set $q(x) = 0$ if $x \in C_{op}$, and $q(x) = 1$ if $x \in C_{fail}$. Subsequently, we assign to e^* the distance:

$$\mathrm{dist}(x, y) = -\log(\max(q(x), q(y))).$$

The following cases exhaust all possibilities. In each case, we explain the suitability of the above function, and outline the construction of the two sets V_{op} and V_{free} (cf. Step 5). Both sets are initially empty.

Case 1: $\max(q(x), q(y)) \neq 0, 1$ (so, $\mathrm{dist}(x, y) \neq \infty, 0$). In this case at least one of x and y is free, and neither node is failed. Failing link (x, y) in G requires failing either x or y. Step 5 selects the node with higher failure probability to include in V_{free}. Here, we note that the distance function associates relatively small values when $\max(q(x), q(y))$ assumes relatively large values. Thus, encouraging the use of nodes with higher failure probability in the identified (D, A_{target})-breach passage.

Case 2: $\max(q(x), q(y)) = 0$ (so, $\mathrm{dist}(x, y) = \infty$). In this case, both x and y operate in C. We distinguish the following cases.

a) Both x and y reach the sink by nodes in C_{op}. Here, it is not possible to fail link (x, y) in G. All such links are deleted from H^* prior to solving the shortest path problem.

b) Exactly one of x or y, say y, reaches the sink by nodes in C_{op}. Failing link (x, y) in G requires disconnecting the other node (node x) from the sink. Disconnecting x incurs a cost incurred by failing possibly additional free nodes to perform the disconnection. We mark such cases by setting $\mathrm{dist}(x, y)$ to a high value, denoted $\mathrm{MAXDIST}$, that is larger than any distance in Case 1. Step 5 adds x to V_{op}.

c) Neither x nor y reaches the sink by nodes in C_{op}. Failing link (x, y) in G can be done by disconnecting either node from the sink. We choose either one of the two nodes, say x. Step 5 adds x to V_{op}. As in the above case, disconnecting x incurs additional cost, so we set $\mathrm{dist}(x, y) = \mathrm{MAXDIST}$.

Case 3: $\max(q(x), q(y)) = 1$ (so, $\mathrm{dist}(x, y) = 0$). In this case, at least one of x or y is failed in C. Thus, link (x, y) is already failed in G. No extra cost is incurred by choosing this link. The distance function captures this aspect by setting $\mathrm{dist}(x, y) = 0$ so as to encourage the shortest path algorithm to choose e^*, as desired.

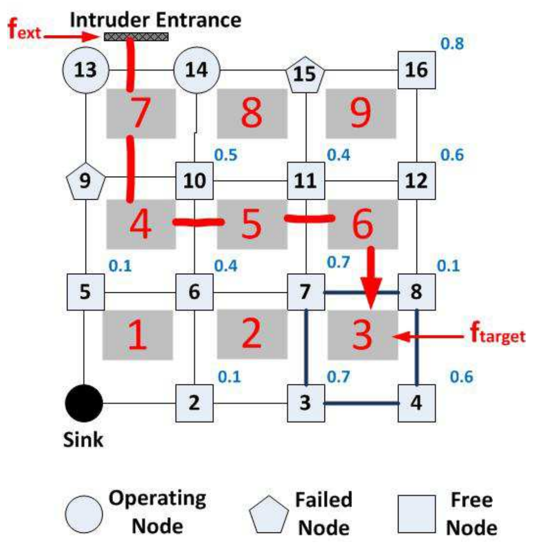


Figure 6.5: Cutset extension example for the BPTA-REL problem

Example. Fig. 6.5 illustrates an instance of the E2C problem where D has one entrance side, $A_{target} = f_3$, and $C = \{(9, fail), (13, op), (14, op), (15, fail)\}$. All other non-sink nodes are free in C. Node operating probabilities appear on the top right. We note the following.

- Steps 1 to 3 identify a shortest (f_{ext}, f_{target})-path in H^*: $(f_{ext}, f_7, f_4, f_5, f_6, f_3)$ on a set E'_{H^*} of links.

- Step 5 sets $V_{op} = \{13\}$, and $V_{free} = \{6, 8, 11\}$. Node 13 is added to V_{op} since link $e^* = (f_{ext}, f_7)$ is associated with link $(x = 13, y = 14)$ in G. Case 2.c applies to link $(13, 14)$ resulting in including either node in V_{op}. Node 6 is added to V_{free} since link $e^* = (f_4, f_5)$ is associated with link $(x = 6, y = 10)$ in G. Case 1 applies to link $(6, 10)$ where $q(6) > q(10)$. Similar argument holds for adding nodes 8 and 11 to V_{free}.

- Step 6 sets $V'_{free} = \emptyset$ since failure of nodes $C_{fail} \bigcup V_{free} = \{9, 15, 6, 8, 11\}$ disconnect the node in $V_{op} = \{13\}$ from the sink. Thus, $V_{free} = \{6, 8, 11\}$.

- Step 7 does not modify V_{free}. Thus, $C_{new} = \{(6, fail), (8, fail), (11, fail)\}$.

■

6.5.3 Correctness

Theorem 6.5 *Function* E2C *computes a feasible extension C_{new} if and only if such extension exists.*

Proof. We first show that if E2C returns $+1$ then $C \bigcup C_{new}$ is a BPTA-REL cutset. In such cases, Step 3 computes a (f_{ext}, f_{target})-path in H^* whose links are denoted E'_{H^*}. The set E'_{H^*} corresponds to a set E'_G of links in G through the $H^*2G()$ relation. Theorem 6.3 ensure that E'_G is a (D, A_{target})-breach passage in G. Thus, failing all links in E'_G allows for undetected intrusion path in G. Step 4 handles the case where each link in E'_G has zero distance (so, $\text{dist}(E'_{H*}) = 0$). As mentioned in Case 3, such link is failed in G, and C is already a cutset. Otherwise, following Step 6, we know that failing all nodes in $C_{fail} \bigcup V_{free}$ guarantees that all links in E'_G fail. Thus, assigning the failed state to all nodes in V_{free} results in a configuration that extends C to a cutset. Step 7 refines V_{free} without violating the above property, as required.

Next, we show that if configuration C admits an extension C_{new} so that $C \bigcup C_{new}$ is a cutset then function E2C returns $+1$ and a possible solution. Denote by V_{breach} the subset of nodes in $C \bigcup C_{new}$ where each node is either failed, or operating but can not possibly reach the sink due to failed nodes in $C_{fail} \bigcup V_{breach}$. Each link of G that is incident to at least one node in V_{breach} is failed in G since it allows for undetected, or unreported crossing. Denote such failed links by E'_G. Since $C \bigcup C_{new}$ is a cutset, it then follows, by definition, that E'_G forms a (D, A_{target})-breach passage. Nodes in C_{new} appear in the input of function E2C as free nodes. So, the function sees nodes in V_{breach} incident to E'_G as either free nodes, or operating nodes the do not reach the sink in the input configuration C. Note that E'_G has no link (x, y) that satisfies Case 2.a (i.e., both x and y operate and reach the sink by nodes in C_{op}). The set E'_G corresponds to a set of links in H^*, denoted E'_{H*}, by the $G2H^*$ relation. Theorem 6.3 ensure that E'_{H^*} contains an (f_{ext}, f_{target})-path in H^*. Deleting links from H^* not in E'_{H^*} (as done in Case 2.a) does not affect such path. The shortest path computation in Step 3 is guaranteed to find such shortest path in H^*. Thus,

$\mathrm{E2C}$ returns $+1$ and a feasible solution, as required. ■

6.6 Numerical Results

In this section we present some of the obtained results. Our experiments use 3 types of grid networks: *grids*, *d-grids* (diagonal grids), and *x-grids* (doubly diagonal grids). We use (x, y) coordinates to describe their structure. A $W{\times}L$ grid has W rows at $y = 0, 1, 2, \ldots, W-1$, and L columns at $x = 0, 1, 2, \ldots, L-1$. A d-grid adds diagonal links of the form $((x, y), (x-1, y-1))$ to grids. An x-grid adds diagonal links of the form $((x-1, y), (x, y-1))$ to d-grids. We assume the use of a routing algorithm that can utilize any link.

6.6.1 Exact $\mathrm{BPTA\text{-}REL}$ solutions

Our algorithms utilize the devised $\mathrm{E2P}$ and $\mathrm{E2C}$ functions to discard configurations that can not be extended to pathsets (or, cutsets). Thus, allowing for efficient $Rel(G)$ computations. Table 6.1 presents the number of configurations generated by the LB to compute exact solutions. For example, a 4×5 grid, and a 5×5 grid have been processed in less than 4 minutes on a personal laptop computer. The computations generated less than 8500 configurations in each case.

Table 6.1: Exact computations for the $\mathrm{BPTA\text{-}REL}$ problem

$W{\times}L$	**Network states**	**Generated configuration**
2×2	2^3	3
2×3	2^5	8
3×3	2^8	14
3×4	2^{11}	52
4×4	2^{15}	187
4×5	2^{19}	1078
5×5	2^{24}	8411

6.6.2 Accuracy of LBs and UBs

To examine the gap between the obtained LBs and UBs, we use $W \times W$ grid networks, $W \in [2, 10]$, with the following parameters: $p(x) = 0.7$ for all non-sink nodes, the sink is at location $(0, 0)$ (bottom left), one entrance side on the top left side is used, and A_{target} is a grid block located at the center of the grid (with top left corner at $(\lceil \frac{W}{2} \rceil - 1, \lceil \frac{W}{2} \rceil))$. Fig. 6.6 shows the obtained results after performing 1000 iterations. As one may expect, the gap increases as the network size increases. This is due to the increased number of configurations that are not accounted for in 1000 iterations. Our experience in analyzing such gap for other reliability problems is that the obtained LBs provide more accurate estimates of the exact result than the obtained UBs. This behaviour is attributed to the existence of pathsets of small size (which results in higher occurrence probability) than cutsets.

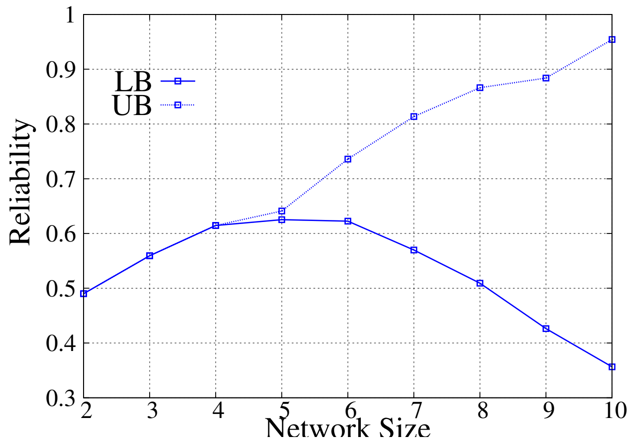


Figure 6.6: Effect of network size on the obtained bounds of the $\mathrm{BPTA\text{-}REL}$ problem

6.6.3 Optimal sink locations

An interesting WSN design problem is to locate the best sink location that maximizes $Rel(G)$. Our results in this section is based on using LBs obtained by performing 1000 iterations in each case. We experimented with both grid networks and random networks. We use a 6×6 grid (d-grid, or x-grid) with one entrance side (on the top left), where A_{target} is a grid block that lies roughly in the middle of the grid (with top left corner at coordinates $(2, 2)$). The sink location is varied on the diagonal: $(0, 0)$, $(1, 1)$, $\cdots$, $(5, 5)$.

Varying node operating probabilities. Fig. 6.7 shows the results obtained using the 6×6 grid mentioned above. The best sink location is found to be at location $(3, 3)$. This is a reasonable outcome as the sink lies midway between A_{target} and the intruder entrance side. Therefore, many barriers can reach the sink using short paths. In addition, the results also show the positive effect of using more reliable nodes.

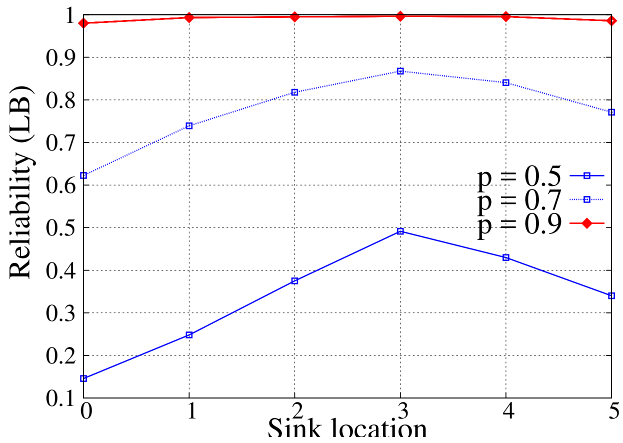


Figure 6.7: Effect of varying sink location on $Rel(G)$ of the BPTA-REL problem for different node operating probabilities

Varying network topology. Fig. 6.8 shows the obtained results using grids, d-grids, and x-grids that are similar to the 6×6 grid network described above. Here, $p(x) = 0.7$ for all non-sink nodes. The results show that the best sink location is still $(3, 3)$. The obtained reliability from using x-grids is better than grids and d-grids. The main reason is that x-grids have significantly more small sized pathsets.

Results on a random network. Fig. 6.9 shows a random network consisting of 36 nodes where $p(x) = 0.7$ for all nodes. The intruder's entrance side, and A_{target} are shown in the figure. The six candidate sink locations are shown as rectangles. The obtained results in table 6.2 show that location 2 is the best sink location. At this location, the sink becomes close to many small sized barriers (e.g., two barriers with only one link each, and one barrier with only two links). In many such barriers, the sink participates actively in sensing. This results in many pathsets having a small total number of nodes each. Thus, many pathsets with high occurrence probabilities exist. Consequently, the obtained LB assumes relatively larger values.

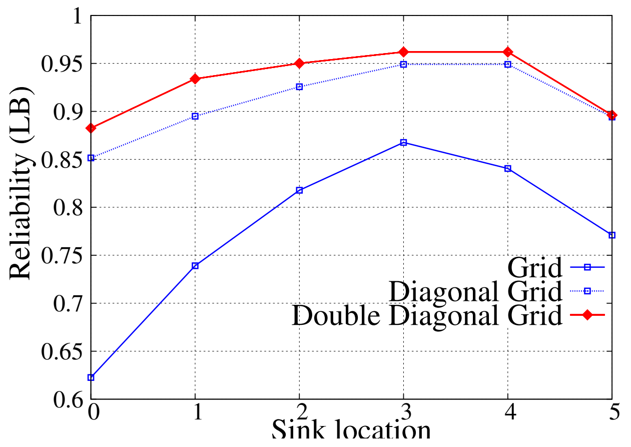


Figure 6.8: Effect of varying sink location on $Rel(G)$ of the BPTA-REL problem for different network topologies

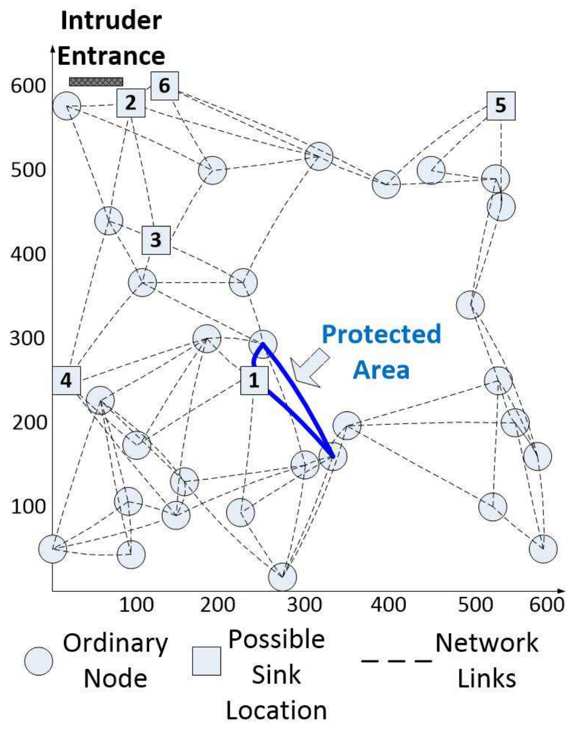


Figure 6.9: A random network instance of the BPTA-REL problem

Table 6.2: Reliability (LB) versus sink location

Sink Location	**Reliability**
1	0.6794
2	0.91
3	0.7
4	0.475
5	0.3848
6	0.637

6.6.4 Effect of D and A_{target} locations

The main objective in this experiment is to study the relationship between the reliability and the locations of intruder entrance sides D and the area A_{target}. The network considered in Fig. 6.10 is 6×6 grid network. The sink lies on the bottom left of the network. The nodes have operating probabilities equal to 0.7. The intruder entrance side is varied over the 20 possible sides of the 6×6 grid. The area to be protected is also varied to be one of the 25 possible square grid cells.

The obtained results of Fig. 6.11 show the following points:

- The least detectable intruder is the intruder entering from entrance 10 aiming to reach area 25. This is predictable since this area lies in the border of the network and only one barrier can detect this intruder also the path to the sink is long.
- As the distance between the target area and the intruder entrance increases, the number of possible barriers increases. Therefore, the reliability increases.
- The nearer the intruder entrance to the sink, the shorter the average path lengths between barriers and the sink node, and the higher is the obtained reliability.
- The most detectable intruder entrance and protected area are (1,5) because of the previous 2 points.

6.7 Concluding Remarks

In this chapter, we develop an efficient method for computing lower and upper bounds for the BPTA-REL problem that takes into account the need to provide joint detection and reporting of intrusion events. Our method utilizes duality among planar graphs to compute effective pathsets and cutsets of a given network. Numerical results show the strengths of the devised method, and its use in tackling an optimum sink placement problem.

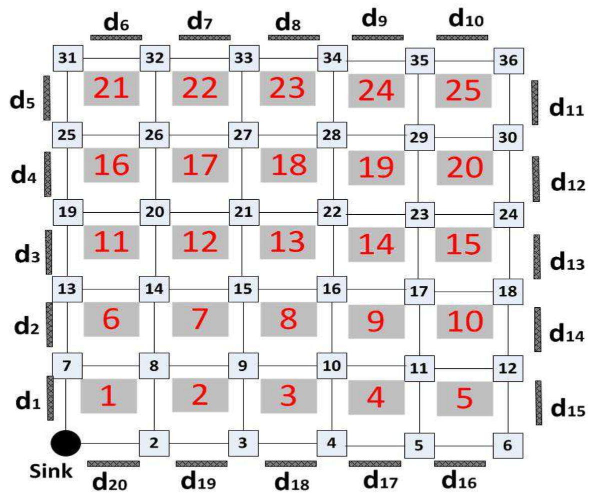


Figure 6.10: A BPTA-REL instance of $6{\times}6$ grid network where $p(x) = 0.7$ for all non-sink nodes

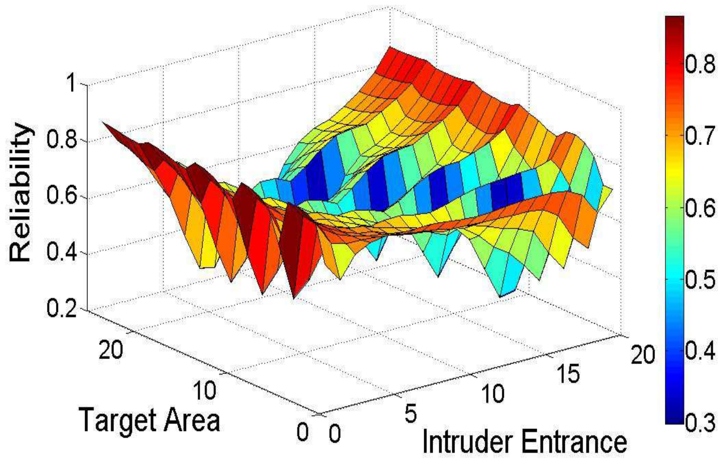


Figure 6.11: Effect of varying intruder entrance side and A_{target} on $Rel(G)$ in $6{\times}6$ grid network where $p(x) = 0.7$ for all non-sink nodes

Chapter 7

Concluding Remarks

In this thesis, we have investigated a class of WSN reliability problems where the nodes collaborate to jointly detect and report to a sink node an unauthorized traversal of a geographic area guarded by the network. This class of event detection problems have received considerable attention in the WSN literature. What distinguishes the research work done in the thesis from many other results in the literature is the focus on quantifying the ability of the network to successfully perform the detection and reporting while taking into consideration the likelihood that any subset of nodes can fail in carrying the required sensing and/or communication tasks.

To this end, the thesis has formalized a number of probabilistic measures suitable for adoption as network reliability measures. The list of formalized problems is as follows.

1. The EXPO problem (Chapter 3) where an intruder path is given as part of the input

2. The BPDREL problem (Chapter 4) where intrusion paths are not given in the input, but the set of possible entry-exit network sides of such paths are specified in the input

3. The DIR-BPDREL problem (Chapter 5) that extends the BPDREL problem to networks with directional communication and sensing nodes

4. The BPTA-REL problem (Chapter 6) where an intruder enters the network from some perimeter point and aims to reach an area of interest (AoI) inside

the network

Our work on the above problems appear in [18,19,22,24]. In addition, we have obtained results in [20,21,23] (not included in the thesis) on some special cases of the above problems as follows.

5. In [20], we develop a dynamic programming algorithm that solves the $\mathrm{BPTA\text{-}REL}$ problem (that deals with an AoI) exactly on grid networks where the sink is located at one of the corners.

6. In [21], we develop a dynamic programming algorithm for the $\mathrm{BPTA\text{-}REL}$ problem for WSNs deployed as concentric rings with the AoI and the sink roughly at the center.

7. In [23], we investigate strategies for packing consecutive cutsets (cf., the section on bounding techniques in Chapter 1) for the BPDREL problem.

7.1 Future Work

In this section, we propose for future work a number of possible problems and directions related to the main thrust of the thesis.

Directions Related to WSNs. First, we note that the thesis has identified 4 extension to a pathset (E2P) problems, and 4 extension to a cutset (E2C) problems. We have obtained results on the computational complexity of some of these problems either by reductions from grid networks (a subclass of *unit disk* graphs (UDGs)), or arbitrary (non wireless) graphs. UDGs are useful models for idealized wireless networks with omni-directional antennas. Complexity results that utilize arbitrary graphs need to be revisited to decide whether the problems remain hard on UDGs. As well, the complexity of the remaining open problems needs to be settled.

Second, the thesis has devised algorithms for the formulated E2P and E2C optimization problems. Not all devised algorithms have shown to have guaranteed performance measures. It is worthwhile investigating if approximation algorithms with bounded approximation ratios can be obtained. Likewise, it is worth developing effective algorithms for special classes of useful WSN topologies.

Third, we recall from Chapter 1 that work on connectivity-based wired networks has derived useful lower and upper bounds from well structured sets of pathsets and cutsets (e.g., structures with the consecutive set property). Such results encourage developing parallel results for WSNs.

Fourth, the importance of directional WSNs motivates investigating many WSN reliability problems on this class. In Chapter 5, we pursued this direction for the BPDREL problem. Similar investigations can be done for the EXPO and the $\mathrm{BPTA\text{-}REL}$ problems.

Fifth, we note that our work thus far has used a sensing intensity function where sensing any point p by a sensor x relies only on the sensor x. Other intensity functions have also been considered in the literature (e.g., *maximum* sensor field intensity, N-*closest* sensor field intensity, and *all-sensor* field intensity). Thus, we propose developing reliability algorithms for different intensity functions under either omnidirectional or directional sensing.

Extensions to Systems Related to WSNs. Currently there is a growing interest in *cyber-physical systems* (CPS) [32,33,44], and *Internet of Things* (IoT) architectures [3,34,52].

Cyber-physical systems embed computing and communication capabilities in many types of physical objects. In [44], the authors describe such systems as physical and engineered systems whose operations are monitored, coordinated, controlled, and integrated by a computing and communication core. Examples of CPS include medical devices, transportation vehicles, factory automation systems, building and environmental control and smart spaces. Compared to the class of WSNs investigated in the thesis, CPS utilize more feedback and actuation mechanisms whereas the WSNs considered in the thesis are modelled as open loop systems with no actuation beyond sending an alarm signal. The existence of feedback and/or actuation makes the development of tractable useful models to quantify the effect of component failure more complex. However, the importance of such systems motivates work in this direction.

The IoT is defined in [52] as a global infrastructure of sensing, computing, storage, and networking platforms. In this vision, IoT objects communicate and interact to provide sensed information and control services for the IoT user applications through a global data management system (cloud). IoT application domains share common grounds with CPS application domains. However, IoT applications rely more on the ability and desirability to collect data from possibly heterogeneous systems spread over a possibly large geographical area. From a networking perspective, an IoT infrastructure spans four layers:

- Layer 1 includes embedded systems and sensors that can be either stationary or mobile.
- Layer 2 provides connectivity using various types of access networks, networking functions implemented at the edge using the *network function virtualization* (NFV) paradigm, and application logic implemented at the edge using the *fog computing* paradigm.
- Layer 3 provides core networking services.
- Layer 4 provides further application logic and business analytics using cloud computing.

Both quality of information (QoI) and quality of service (QoS) metrics are used by IoT applications to take decisions and provide feedback to end users, and layer 1 devices.

Currently, research work is being conducted on various design aspects of IoT systems including identifying useful QoI and QoS metrics for selected IoT applications, identifying layer 1 resources that can be adaptively managed and reconfigured to satisfy target performance levels, and designing methodologies to manage and reconfigure the available resources under different operational constraints. In addition to the above directions, we propose to formalize and seek solutions to problems that quantify the performance of IoT systems under different scenarios of component malfunction.

Bibliography

[1] H. AboElFotoh and C. Colbourn. Computing 2-terminal reliability for radio-broadcast networks. *IEEE Transactions on Reliability*, 38(5):538 – 555, August 1989.

[2] H.M.F. AboElFotoh, E.S. Elmallah, and H.S. Hassanein. A flow-based reliability measure for wireless sensor networks. *International Journal of Sensor Networks*, 2(5/6):311 – 320, December 2007.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347 – 2376, Fourthquarter 2015.

[4] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

[5] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial *k*-trees. *Discrete Applied Mathematics*, 23(1):11–24, 1989.

[6] J. Bachrach and C. Taylor. *Handbook of Sensor Networks: Algorithms and Architectures*. Wiley-Interscience, 2005.

[7] C. Bettstetter. On the minimum node degree and connectivity of a wireless multihop network. In *The 3rd ACM International Symposium on Mobile Adhoc Networking & Computing (MobiHoc)*, pages 80 – 91, Lausanne, Switzerland, June 2002.

[8] S. Bhatti and J. Xu. Survey of target tracking protocols using wireless sensor network. In *The fifth International Conference on Wireless and Mobile Communications (ICWMC)*, pages 110 – 115, August 2009.

[9] H. L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In *Proceeding 15th International Colloquium on Automata, Languages and Programming*, volume 317 of *Lecture Notes in Computer Science*, pages 105–118. Springer Berlin Heidelberg, 1988.

[10] J. A. Bondy and U.S.R. Murty. *Graph theory*. Graduate texts in mathematics. Springer, New York, London, 2007.

[11] A. Boukerche, H. A. B. Oliveira, E.F. Nakamura, and A.A.F. Loureiro. Localization systems for wireless sensor networks. *IEEE Wireless Communications*, 14(6):1536 – 1284, December 2007.

[12] A. Boukerche, H. A. B. F. Oliveira, E. F. Nakamura, and A. A. F. Loureiro. *Algorithms and Protocols for Wireless Sensor Networks*, pages 307–340. John Wiley and Sons, Inc., 2008.

[13] A. Brandstädt, V. Bang Le, and J. P. Spinrad. *Graph classes: a survey*, volume 3. SIAM, Philadelphia, PA, USA, 1999.

[14] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. Saluja. Sensor deployment strategy for detection of targets traversing a region. *Mobile Networks and Applications*, 8(4):453 – 461, August 2003.

[15] C. J. Colbourn. *The Combinatorics of Network Reliability*. Oxford University Press, 1987.

[16] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw Hill, MIT Press, third edition, 2009.

[17] O. Demigha, W. Hidouci, and T. Ahmed. On energy efficiency in collaborative target tracking in wireless sensor network: A review. *IEEE Communications Surveys and Tutorials*, 15(3):1210 – 1222, 2013.

[18] M. Elmorsy and E. Elmallah. Breach path to target area detection reliability in wireless sensor networks. In *the 39th IEEE Conference on Local Computer Networks (LCN)*, pages 253 – 261, Edmonton, Canada, September 2014.

[19] M. Elmorsy and E. Elmallah. On pathsets and cutsets of a wireless sensor network surveillance problem. In *the IEEE International Conference on Communications (ICC)*, pages 459 – 465, Sydney, Australia, June 2014.

[20] M. Elmorsy and E. Elmallah. Guarding an area of interest in sensor grids with unreliable nodes. In *the IEEE International Conference on Communications (ICC)*, pages 6456 – 6462, London, UK, June 2015.

[21] M. Elmorsy and E. Elmallah. Reliable surveillance in ring deployed wireless sensor networks. In *the 40th IEEE Conference on Local Computer Networks (LCN)*, pages 133 – 140, Florida, USA, October 2015.

[22] M. Elmorsy and E. Elmallah. Breach path reliability for directional sensor networks. In *the 41st IEEE Conference on Local Computer Networks (LCN)*, pages 371 – 379, Dubai, UAE, November 2016.

[23] M. Elmorsy and E. Elmallah. Packing of cutsets for a breach path detection problem. In *the IEEE International Conference on Communications (ICC)*, pages 1 – 7, Kuala Lumpur, Malaysia, May 2016.

[24] M. Elmorsy, E. Elmallah, and H.M.F. AboElFotoh. On path exposure in probalistic wireless sensor networks. In *the 38th IEEE Conference on Local Computer Networks (LCN)*, pages 433 – 440, Sydney, Australia, October 2013.

[25] K. Fan and P. Sinha. Distributed online data aggregation for large scale sensor networks. In *5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, pages 153 – 162, October 2008.

[26] K. W. Fan, S. Liu, and P. Sinha. Structure-free data aggregation in sensor networks. *IEEE Transactions on Mobile Computing*, 6(8):929 – 942, August 2007.

[27] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. In-network aggregation techniques for wireless sensor networks: A survey. *IEEE Wireless Magazine*, 14(2):70 – 87, April 2007.

[28] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman, San Francisco, 1979.

[29] G. Han, H. Xu, T. Q. Duong, J. Jiang, and T. Hara. Localization algorithms of wireless sensor networks: a survey. *Telecommunication Systems*, 52(4):2419 – 2436, 2013.

[30] D. D. Harms, M. Kraetzl, C. J. Colbourn, and J. S. Devitt. *Network Reliability Experiments with a Symbolic Algebra Environment*. CRC Press, 1995.

[31] N. Katneni, V. Pandit, H. Li, and D. P. Agrawal. Hybrid gaussian-ring deployment for intrusion detection in wireless sensor networks. In *IEEE International Conference on Communications (ICC)*, pages 549 – 553, Rome, Italy, June 2012.

[32] S. K. Khaitan and J. D. McCalley. Design techniques and applications of cyberphysical systems: A survey. *IEEE Systems Journal*, 9(2):350 – 365, June 2015.

[33] K. D. Kim and P. R. Kumar. Cyber-physical systems: A perspective at the centennial. *Proceedings of the IEEE*, 100 (Special Centennial Issue):1287 – 1308, May 2012.

[34] S. Kraijak and P. Tuwanut. A survey on IoT architectures, protocols, applications, security, privacy, real-world implementation and future trends. In *the 11th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pages 1 – 6, Sept 2015.

[35] D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed. Detection, classification and tracking of targets. *IEEE Signal Processing Magazine*, 19(2):17 – 29, August 2002.

[36] B. Liu and D. Towsley. A study of the coverage of large-scale sensor networks. In *the IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 475 – 483, October 2004.

[37] L. Liu, X. Zhang, and H. Ma. Minimal exposure path algorithms for directional sensor networks. In *IEEE Global Telecommunications Conference (GLOBECOM)*, 2009.

[38] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *IEEE INFOCOM*, pages 1380 – 1387, 2001.

[39] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad-hoc sensor networks. In *The 7th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 139 – 150, Rome, Italy, 2001.

[40] M. Naderan, M. Dehghan, and H. Pedram. Mobile object tracking techniques in wireless sensor networks. In *International Conference on Ultra Modern Telecommunications & Workshops (ICUMT)*, pages 1 – 8, October 2009.

[41] E. Onur, C. Ersoy, H. Delic, and L. Akarun. Surveillance with wireless sensor networks in obstruction: Breach paths as watershed contours. *The International Journal of Computer and Telecommunications Networking*, 54(3):428 – 441, February 2010.

[42] M. D. Penrose. On k-connectivity for a geometric random graph. *Wiley Random Structures and Algorithms*, 15(2):145 – 164, July 1999.

[43] J. Provan and M. Ball. Computing network reliability in time polynomial in the number of cuts. *Operations Research*, 32(3):516 – 526, June 1984.

[44] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: The next computing revolution. In *the 47th ACM/IEEE Design Automation Conference (DAC)*, pages 731 – 736, June 2010.

[45] M. H. Shazly, E. Elmallah, and J. Harms. An approach for bounding breach path detection reliability in wireless sensor networks. In *the IEEE 37th Conference on Local Computer Networks (LCN)*, pages 585 – 592, Clearwater, Florida, USA, October 2012.

[46] M. H. Shazly, E. Elmallah, and J. Harms. On breach path detection reliability of wireless sensor grids. In *The 21st International Conference on Computer Communications and Networks (ICCCN)*, pages 1 – 7, Munich, Germany, August 2012.

[47] D. R. Shier. *Network Reliability and Algebraic Structures*. Oxford University Press, 1991.

[48] D. Smith and S. Singh. Approaches to multisensor data fusion in target tracking: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 18(12):1696 – 1710, December 2006.

[49] H.J. Strayer and C.J. Colbourn. Consecutive cuts and paths, and bounds on k-terminal reliability. *Networks*, 25(3):165 – 175, May 1995.

[50] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189 – 201, 1979.

[51] V. V. Vazirani. *Approximation Algorithms*. Springer, 2003.

[52] O. Vermesan and P. Friess. *Internet of Things: From Research and Innovation to Market Deployment*. River Publishers series in communications. River Publishers, 2014.

[53] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583 – 598, June 1991.

[54] Y. Xu, J. Winter, and W.C. Lee. Prediction-based strategies for energy saving in object tracking sensor networks. In *the IEEE International Conference on Mobile Data Management (MDM)*, pages 346 – 357, January 2004.