Path Exposure Reliability Problem in Wireless Sensor Networks with Energy Replenishment

by

Abdulsalam Salem Saeed Basabaa

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

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

Advances in sensor devices and harvesting technologies have enabled the design of small, inexpensive, and low-power energy harvesting wireless sensor networks (EH-WSNs) that can be used to collect data and provide reliable monitoring in many applications such as surveillance applications (e.g., intrusion detection). However, in such EH-WSNs, fluctuations in nodes stored energy levels arise as a common aspect and present a challenge in achieving prolonged network uptimes.

The thesis considers EH-WSNs where fluctuations in a node's stored energy affect primarily its transmission range, and studies a class of intrusion detection problems where an unauthorized traversal aims at crossing a geographical area guarded by an EH-WSN. The main objective is to develop methodologies to quantify the likelihood that an EH-WSN whose nodes are subject to energy fluctuations can provide simultaneous detection and reporting of unauthorized traversal along a given path to a sink node.

The thesis pursues this objective in the context of investigating a path exposure reliability problem that calls for computing the probability that the collaborative work of all nodes in a given EH-WSN succeeds in detecting and reporting an intrusion along a given path. We show that the problem is computationally intractable, and the thesis focuses on developing iteratively and non-iteratively improvable algorithms that utilize special network structures, called pathsets and cutsets, for driving lower and upper bounds on the exact solutions. We conclude with some possible future research problems.

Preface

This thesis is the original work done by Abdulsalam Basabaa under supervision of Professor Ehab Elmallah. Publications [8–12] contain original contributions mentioned in Chapters 3 to 7.

The paper in [12] has received the Best Student Paper Award at the 11^{th} *International Conference on Sensor Networks (SENSORNETS)* held from February 7 - 8, 2022.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Ehab Elmallah for his guidance and support during my PhD work.

Sincere gratitude are also to my thesis examination committee members: Professors Janelle Harms, Omid Ardakanian, Ioanis Nikolaidis, Davood Rafiei, and Fayez Gebali for the valuable comments and suggestions.

I would like also to acknowledge the financial support from Hadhramout Establishment for Human Development (HEHD) scholarship, Yemen.

Last but not the least, I would like to give special thank to my wife and children for being the source of joy and happiness.

Contents

List of Tables

List of Figures

List of Acronyms and Notation

CSP	Consecutive set of pathsets
DirEXPO-RU	Directional path exposure with range uncertainty problem
E2C	Extension to a cutset problem
E2C-BFS	Extension to a cutset using breadth-first search
E2C-MaxFlow	Extension to a cutset using a max-flow algorithm
E2P	Extension to a pathset problem
EH-WSNs	Energy harvesting wireless sensor networks
$\text{Expo}(G, p, P, k_{req})$	The probability that the intrusion path P can be jointly detected and reported to the sink node by at least k_{req} sensing nodes. Also, abbreviated as $\text{Expo}(G, p, k_{req})$ and $\text{Expo}(G, k_{req})$
EXPO-RU	Path exposure with range uncertainty problem
HWAS	Half-width angle selection problem
LB	Lower bound
NDP	Node disjoint pathset
$p_s(x)$ or $p(x, s)$	The probability that node x is in state s, e.g., $s \in \{full, reduced, fail\}$
SOC	State-of-charge (of a rechargeable battery or a supercapacitor)
UB	Upper bound
USP	Unrestricted set of pathsets

Chapter 1

Introduction

The main theme of the thesis is on analyzing the ability of a wireless sensor network (WSN) whose nodes utilize different means of energy harvesting from the environment (e.g., solar power) to successfully achieve a well defined task. In the introductory part of this chapter (Section 1.1), we motivate this research direction, and give more details on the scope of the thesis. A general mathematical model that can be used to formalize many problems of interest associates a discrete probability state space with some elements (e.g., nodes and/or links) of a given network, giving rise to the concept of a *probabilistic network* model. Many important concepts and algorithms have evolved in recent years on the use of probabilistic network models to analyze various network reliability measures. To start, we review some of the needed concepts in Section 1.2. In Section 1.4, we review some existing work for the class of path exposure problems in WSNs. Section 1.5 outlines the main contributions and structure of the thesis.

1.1 Introduction

The field of wireless sensor networks (WSNs) has been an active research and development field in networking since around 1999 (see, e.g., the survey in [4]). Typically, a WSN is composed of a dense deployment of low cost, low power miniaturized devices (called sensor nodes) that integrate sensing, computation, and communication. WSNs continue to receive significant attention in research and industry for providing solution platforms in may areas of appli-

cations (e.g., environment monitoring, structural health monitoring, human health monitoring, surveillance and target tracking, etc.). Early generations of WSNs are designed to utilize non-rechargeable batteries. Research done on such networks has resulted in a rich body of literature on various energy conservation approaches that collectively aim at reducing the energy consumed by the whole network during operation. The objective is to maximize the lifetime of a WSN while satisfying acceptable levels of performance. Among such approaches, we mention the class of topological management (also called efficient scheduling) algorithms surveyed in [54], the class of topology control mechanisms surveyed in [43], the class of power-aware routing algorithms surveyed in [53], and the class of power-aware MAC protocols surveyed in [55]. Surveys spanning different approaches appear in [5, 40].

More recently, with the advances in designing efficient solar panels, micro wind tribunes, and wireless charging technologies, more research work has devoted attention to energy replenishment in terrestrial WSNs to achieve increased network uptime. Research on energy replenishment in WSNs straddles two broad directions. The first broad direction relies on harvesting energy from the environment such as using solar power, vibrations, or thermoelectric effects. The second broad direction for energy replenishment relies on using wireless energy transmission.

The scope of the thesis work is on investigating some reliability aspects of energy harvesting WSNs (EH-WSNs) deployed to detect events that occur in a given geographical area. The goal is to be able to quantify the likelihood that an EH-WSN with nodes subject to energy fluctuations can successfully achieve its task. To this end, our research work aims at identifying useful EH-WSN models, formalizing useful EH-WSN reliability problems, analyzing the complexity of the resulting problems, and developing efficient algorithms to solve the problems. The algorithms developed in the thesis aim at enabling a network designer to differentiate among different candidate topologies, where nodes in each network employ an energy management unit that controls the maximum transmission range of a node. The differentiation relies on using a reliability metric that captures that collective work of all nodes in the network

to achieve a given task. Network topologies are assumed to be given as input. The developed algorithms, however, do not provide a direct means of controlling the nodes during daily operation. Thus, our research direction is similar to the research work done in [19, 20, 47]. However, we focus here on the class of EH-WSNs. In the next section, we review some of the needed concepts and results in this direction.

1.2 Concepts in Network Reliability Analysis

In this section, we introduce some definitions and concepts used in our work. The concepts have been used in the early work of [16] and [7] and the references therein. For simplicity, we introduce the concepts in the context of a fundamental graph problem, called the *2-terminal* network reliability problem (denoted Rel_2) mentioned in [16] and [7]. We note that, although our main problem that is dealt with in the thesis is different from the Rel_2 problem, many of the concepts introduced in this section are used in the thesis.

In the Rel_2 problem, we are given an undirected graph $G = (V, E, p)$ on a set V of nodes and a set E of edges, with two distinguished terminal nodes denoted s and t (see, e.g., Fig 1.2.1). Edges are assumed to operate/fail independently of each other. We denote by $p(e)$ (respectively, $q(e) = 1 - p(e)$) the probability that edge $e \in E$ operates (respectively, fails). Thus, we use a *2-state* probability distribution to model the behaviour of each edge. The Rel_2 problem asks for computing the probability that after an event of a random failure, G is in a state where s can reach t by a path of operating edges.

The Rel_2 problem is one of many problems on both wired and wireless networks that have been analyzed using similar models where we use a discrete state space to model the behaviour of each network element (e.g., node and/or edge). We now introduce the following concepts.

1.2.1 Probabilistic Graphs, Pathsets, and Cutsets

The graph $G = (V, E)$ with the probabilities $\{p(e) : e \in E\}$ is called a probabilistic graph, denoted $G = (V, E, p)$. The Rel_2 problem on $G = (V, E, p)$ with

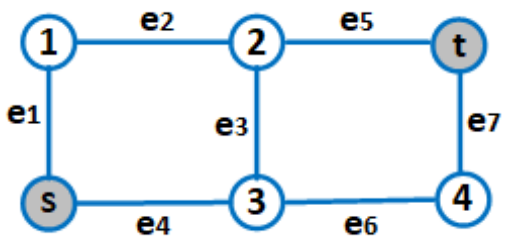


Figure 1.2.1: An example of the 2-terminal network reliability problem

the two distinguished terminals s and t is denoted $Rel_2(G, s, t)$. When each edge in E is assigned a state in $\{operate, fail\}$, we obtain a *network state* of G. We use (e, s_e) to denote the event that edge e is in state $s_e \in \{operate, fail\}$. A network state S can then be written as $S = \{(e, s_e) : e \in E\}$. Likewise, we use $p(e, s_e)$ to denote the probability that edge e is in state s_e. Thus, when edges operate independently of each other, the probability that G is in network state S is $\Pr(S) = \prod_{(e,s_e)\in S} p(e, s_e)$. We also need the following definitions.

- A **network configuration** C is a partial network state where some edges (but not necessary all) are assigned a state. The remaining unassigned edges are *free* in C. For the Rel_2 problem, a network state or configuration is operating if s can reach t in S. Otherwise, C is a failed configuration. When edges operate independently of each other, we get $\Pr[\text{C occurs}] = \prod_{(e,s_e)\in C} p(e, s_e)$.

- A **pathset** is an operating network configuration. A *minpath* is minimal pathset. For example, in Figure 1.2.1, $S = \{(e_4, operate), (e_3, operate), (e_5, operate)\}$ is a pathset (in fact, S is a minpath) of the Rel_2 problem.

- A **cutset** is a failed configuration even if all unassigned (free) elements operate. A *mincut* is minimal cutset. For example, in Figure 1.2.1, $S = \{(e_1, fail), (e_4, fail)\}$ is a cutset (in fact, a mincut) of the Rel_2 problem.

An Exact Algorithm: Given an instance (G, s, t) of the Rel_2 problem, an exact algorithm may proceed by generating the set $OP(G)$ of all distinct operating network states (i.e., pathsets of size $|E|$ each), and computing

$Rel_2(G, s, t) = \sum(\Pr(S) : S \in OP(G))$. The worst case running time of such complete state enumeration algorithm, however, is $O(2^{|E|})$ time.

1.2.2 Computational Complexity

In [16], all formulated reliability problems (a total of 6 problems) have been shown to be #P-complete on arbitrary graphs even in cases of equal operating probabilities. The classes of #P-hard and #P-complete have been introduced in [52] to capture the computational complexity of *counting* problems (as opposed to NP-hard and NP-complete decision problems).

1.2.3 Algorithmic Approaches

To cope with the apparent computational intractability of many known reliability problems on arbitrary networks, researchers have considered a number of approaches to obtain useful algorithms (see, e.g., [7, 16]). Of these approaches, we mention the following.

1. **Exact algorithms on special classes of graphs**:

   WSNs that are deployed deterministically may have simple graph structures (e.g., trees) that allow a reliability problem to be solved in polynomial time. Examples of work in this direction includes work utilizing interval graphs [2], partial k-trees (for any fixed $k \geq 1$) [1], and grid networks [18].

2. **Obtaining bounds from edge-disjoint pathsets and cutsets:**

   Given a set of edge-disjoint pathsets $\{p_1, p_2, ..., p_r\}$, a lower bound on the network reliability can be computed as in formula (1.2.1):

$$\Pr[\text{at least one pathset } p_i \text{ occurs}] = 1 - \prod_{i=1}^{r}(1 - Pr(p_i)) \qquad (1.2.1)$$

   Likewise, given a set of edge-disjoint cutsets $\{c_1, c_2, ..., c_r\}$, we can obtain an upper bound as in formula (4.3.2):

$$\Pr[\text{None of the cutsets occur}] = \prod_{i=1}^{r}(1 - Pr(c_i)) \qquad (1.2.2)$$

3. **Obtaining bounds from consecutive pathsets and cutsets:**

   In [44, 45], the author has introduced the concept of consecutive sets as follows: a sequence $Q^* = (q_1, q_2, ...q_r)$ of r *sets* satisfies the *consecutive* set property **if and only if** for every element x that belongs to sets q_i, q_j, $i \leq j$, x also occurs in every set q_k, where $i < k < j$. In [44], the author has shown an efficient algorithm to compute an UB from a given set of consecutive cutsets. Later, in [50], the authors have shown an efficient algorithm to compute a LB from a given set of consecutive pathsets. Note that, by definition, any edge-disjoint set satisfies the consecutive sets property. Thus LBs and UBs from consecutive sets can achieve strict improvements over bounds from edge-disjoint sets.

4. **Obtaining bounds from statistically-disjoint pathsets and cutsets:**

   The above bounding approaches using edge-disjoint and consecutive pathsets (or cutsets) are called *non-iteratively improvable* approaches. A *non-iteratively improvable* approach produces a single LB or UB from a given set of edge-disjoint or consecutive configurations (pathsets or cutsets). In contrast, the bounds described here are *iteratively improvable*. That is, the obtained bounds improve as we allow the algorithm to perform more iterations. The idea is to generate a set of configurations (pathsets or cutsets) whose occurrence probability can be added to each other. In more detail, we call two configurations C_i and C_j *statistically-disjoint* (**s-disjoint**, for short) if they have at least one common node say, x, that is assigned to two different states in C_i and C_j. Thus, $\Pr(C_i) + \Pr(C_j)$ is the probability that at least one of C_i and C_j occurs. Consequently, if $\{p_1, p_2, ..., p_r\}$ is a set of pairwise s-disjoint pathsets then a LB can be computed using formula (4.3.1):

$$LB = \sum_{i=1}^{r} Pr(p_i) \tag{1.2.3}$$

   Likewise, if $\{c_1, c_2, ..., c_r\}$ is a set of pairwise s-disjoint cutsets then an

UB can be computed using formula (1.2.4):

$$UB = 1 - \sum_{i=1}^{r} Pr(c_i) \tag{1.2.4}$$

To utilize this approach, one needs an algorithm that can systematically generate a maximal set of pairwise s-disjoint configurations in the network. A framework called *factoring* is discussed in [16] (and the references therein). In [20] the authors adapt the framework to generate s-disjoint configurations. The factoring method generates s-disjoint configurations iteratively in a tree-like manner starting from the empty configuration as the root of the tree. At each iteration, the method selects a configuration C that is perceived to be most promising (has highest occurrence probability). Then the configuration C is used to generate new pairwise s-disjoint configurations that are added to the tree. Additional steps can be added to the factoring method in order to obtain a set of s-disjoint pathsets (or cutsets) and compute a LB (respectively, UB) from the generated s-disjoint pathsets (respectively, cutsets). In the additional steps to compute a LB, in each iteration the selected configuration C is tested for extensibility to a pathset. If the configuration C is not extensible, it is labelled as a bad configuration. Else, if it is extensible then it is processed by **(a)** storing the obtained pathset in the tree, and **(b)** generating child configurations that are s-disjoint from all other configurations in the tree.

1.3 Node Model

We assume a standard sensor node (also called sensor mote) equipped with one (or more) sensing device, a data processing unit, a wireless transmitter/receiver device, an energy storage unit (with, e.g., a rechargeable battery and/or supercapacitor), and an energy management unit (EMU) that controls the activity of the node. We also assume that the network topology (a directed graph $G = (V, E)$ on a set of nodes V, and a set of directed links E) is given as input. So, an arc $(x, y) \in E$ if x's transmission power is sufficient

to satisfy the required signal-to-noise-plus-interference ratio threshold at the receiving node y. In addition, if there exists a directed (x, y)-*Path* in some state of the network G, then it is assumed that the routing algorithm used by the network is able to detect and use such a path. Throughout the thesis, we assume that the amount of energy left in a node's energy storage system (e.g., a rechargeable battery and/or supercapacitor) can be estimated. Such amount of energy is also referred to as the state-of-charge (SOC). Currently there is a growing body of research on practical and accurate methods of estimating the state-of-charge of lithium-ion batteries and supercapacitors (c.f., [13, 41, 42, 56]).

1.4 Work on WSNs Reliability for Path Exposure Problem

The path exposure problem is a well-known problem in conventional WSNs. The problem asks for computing the probability of detecting a moving target along a given path over a period of time through a WSN field (see, e.g., [15, 32]). In [15], the authors analyze target detection probability using different data fusion models, and present a method to minimize network deployment cost while achieving desired exposure levels. In [32], the authors present a model that utilizes energy sensed from a moving target over a period of time. The model is then used to find a path of minimal exposure using a shortest path algorithm. Subsequently, authors in [28] formalize a minimal path exposure problem for WSNs employing directional communication. They develop a directional sensing model used to define two weighted graphs that are used to reduce the minimal path exposure problem into two discrete geometry problems. Their results include developing two approximation algorithms to solve the problem. Also the work of [20], where the authors formalize a path exposure problem, called EXPO, on conventional WSNs for surveillance applications where nodes are subject to communication and/or sensing failure.

In contrast to previous work, the thesis considers a new variant of path exposure problem in EH-WSNs where at any instant different nodes have dif-

ferent transmission ranges based on their energy levels.

1.5 Thesis Contributions and Organization

The main theme of the thesis is on quantifying the likelihood that an EH-WSN whose nodes are subject to energy fluctuations can provide simultaneous detection and reporting of unauthorized traversal along a given path. The thesis pursues this objective in the context of investigating a path exposure reliability problem under range uncertainty that calls for computing the probability that the collaborative work of the all nodes in EH-WSN succeeds in detecting and reporting an intrusion along a given path. Here, we summarize the main contributions of the thesis as follows.

1. In Chapter 3, we formulate a *path exposure with range uncertainty* problem, denoted EXPO-RU, where fluctuation in nodes energy levels over time affects mainly their transmission ranges, and nodes can fail independently of each other. We show that the EXPO-RU is #P-hard. In addition, we formulate a problem called the *extension to pathset* problem, denoted E2P, for the EXPO-RU problem. We propose an efficient heuristic algorithm to solve the E2P problem and use the devised algorithm to design both iteratively and non-iteratively improvable LB algorithms for the EXPO-RU problem.

2. In Chapter 4, we extend our work in Chapter 3 by formulating an *extension to cutset* problem, denoted E2C, for the EXPO-RU problem. We propose two efficient heuristic algorithms to solve the E2C problem, and use the devised algorithms to design both iteratively and non-iteratively improvable UB algorithms for the EXPO-RU problem.

3. Chapter 5 considers two general types of reliability problems on probabilistic graphs where each node can be independently in one of possible node state (such as the EXPO-RU problem). Each type has two problems defined on pathsets and cutsets, respectively. In the cutsets version of the first problem type, we are given a sequence Q of network cutsets,

and we want to compute an UB on the exact solution by computing the probability that no cutset in Q occurs. The pathset version is defined similarly to compute a LB. We develop a dynamic programming approach to compute such probability, assuming that node state satisfy a coherence property. In general, the running time of the devised approach grows exponentially with the size of the input set Q. We show, however, that when Q is a disjoint set of cutsets, or a consecutive set, the running time is comparable to the running time of an algorithm that is aware that the input has this property.

In the cutset version of the second problem type, we are given a disjoint set Q of cutsets, and we seek to extend it to a set of consecutive cutsets that exploits the multi-state node model.

4. In Chapter 6, we consider a variant of EXPO-RU problem where nodes are equipped with directional communication devices, denoted DirEXPO-RU. The DirEXPO-RU problem is to quantify the ability of a network to detect and report traversal along a given path. A problem that arises in managing the network resources to maximize this reliability measure is to adjust the transmission beam width of each node, where nodes beam centers are given as input. We formalize a half-width angle selection problem, denoted HWAS, and propose two configuration approaches to adjust the transmission beam width of each node in the network. The proposed approaches are presented in a unified way. The obtained numerical results compare the obtained results with omnidirectional transmissions.

Table 1.1 summarizes the main contributions of the thesis. As mention above, *non-iteratively improvable* LB or UB algorithms refer to a class of bounding algorithms that produce a single bound on each input. Such algorithms are typically fast, but they compute just one value for each input. In contrast, *iteratively improvable* LB or UB algorithms can achieve exact solutions if allowed to execute until termination. The results have been published in [8–12]. Reference [12] has received the Best Student Paper Aware in

Table 1.1: A summary of thesis contributions

Chapter	**Problem**	**Subproblem**	**Obtained results**			
			Non-iteratively improvable algorithms		**Iteratively improvable algorithms**	
			LB	**UB**	**LB**	**UB**
3 [8]	EXPO-RU	E2P	✓		✓	
4 [10, 11]		E2C		✓		✓
5 [9, 10]		CS_extend	✓	✓		
6 [12]	DirEXPO-RU	HWAS	Configuring node's beam width			

SENSORNETS 2022.

1.6 Concluding Remarks

In this chapter, we have motivated research on developing efficient algorithms to assess the reliability of EH-WSNs. We have also reviewed some fundamental concepts and algorithmic approaches useful in obtaining lower and upper bounds on exact solutions. The, we have discussed the main contributions of the thesis. In the next chapter, we review some recent research work in the area of EH-WSNs.

Chapter 2

Literature Review on EH-WSNs

The scope of the thesis work is on modelling and analyzing the reliability of WSNs that utilize energy replenishment in their operation, with a particular emphasis on the class of energy harvesting WSNs (EH-WSNs). Most of the networking research work done to date on EH-WSNs concerns the link layer (MAC protocols), the network layer (routing), and cross-layer energy efficiency protocols. Understanding the work done on routing algorithms, however, is important for understanding network failure modes, and contributes to the ability of developing accurate network models, node state models, and reliability evaluation measures. To this end, we review some basic concepts in the design of EH-WSNs in Section 2.1, followed by a review of some qualitative aspects of some recent EH-WSN routing algorithms in section 2.2, and some mathematical analysis aspects of these routing algorithms in Section 2.3.

2.1 Introduction

As mentioned in Chapter 1, WSNs that utilize energy replenishment has attracted much research over more than one decade now (see, e.g., the surveys in [3, 51]). In this section, we give a brief overview on the design and operation of such networks.

Energy Sources: In [3], the authors classify the sources of harvestable energy into 3 broad classes:

- Radiant sources such as sunlight and radio frequency (RF)

- Mechanical sources such as vibrations, wind, human body motion, and water flow
- Thermal sources that utilize temperature difference between two conducting surfaces where the amount of generated energy depends on the temperature gradient between the surfaces

Table 2.1 (due to [3]) summarizes some harvesting technologies and power densities associated with the above sources.

Table 2.1: Energy sources and their corresponding power densities (due to [3])

Energy Source	Types	Harvesting Method	Power Density
Radiant	solar	solar cell (indoor)	$10\mu W/cm^2$
		solar cell (outdoor)	$15mW/cm^2$
	RF	electromagnetic conversion	$0.1\mu W/cm^2$ (GSM)
		electromagnetic conversion	$0.01\mu W/cm^2$ (WiFi)
Mechanical	wind flow	electromagnetic conversion	$16.2\mu W/cm^3$
	motion	piezoelectric	$330\mu W/cm^3$
Thermal	body heat	thermoelectric	$40\mu W/cm^2$

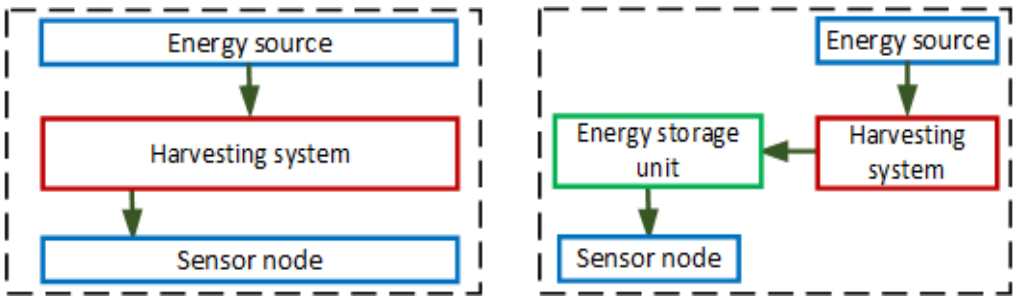
Of the above energy sources, outdoor solar energy (with power density of $15mW/cm^2$) has been shown to provide adequate source to power miniaturized sensor nodes. Solar harvesting technology has allowed the construction of real life EH-WSN prototypes such as the Helliomote prototype of [39], and the Prometheus prototype of [25]. Solar energy and wind energy harvesting are subject to weather conditions. Early work on EH-WSNs has explored the use of weather forecasts to improve a network's ability to predict the amount of energy that can be harvested during relatively long time (see, e.g., [46]). In [46], suitable energy models have been developed for the specific solar panel and wind tribune used in the conducted empirical study. Wireless energy transmission has also received extensive attention for powering WSNs. Early real life prototypes for wireless energy transmission include the work of [37] where the authors have built a proof-of-concept prototype composed of a mobile charger (MC), a mobile robot carrying a wireless power charger, and a network of sensor nodes. Wireless charging at $45mW$ when the charger

is located $10cm$ away from a sensor is reported. Surveys on wireless energy transmissions appear in [29, 30, 48].

Energy Harvesting architectures: In [51], the authors note the following two types of architectures for using the harvested energy:

a) Harvest-use architecture: Here, the harvested energy is used directly to supply the sensor node with the required energy. This architecture does not require an extra unit to store harvested energy but the energy generated by the harvester has to be continuously above the energy required to operate the sensor node, otherwise, the node will shut down.

b) Harvest-store-use architecture: In such an architecture, an energy storage unit (battery and/or super-capacitor) is used to power a sensor node when either a harvesting opportunity does not exist (i.e., during the night for solar energy), or a sensor node needs more energy to perform more functionality. The energy storage unit is particularly useful when the harvested energy rate is more than the required energy by a sensor node.



(a) Harvest-Use (b) Harvest-Store-Use

Figure 2.1.1: Types of energy harvesting architectures (adapted from [51])

The energy storage unit can be implemented using either a single storage device (primary storage), or a double storage device (primary and secondary), where the secondary storage device may work as a backup for situations when the primary storage is exhausted. The energy storage device can be a rechargeable battery and/or super-capacitor. A rechargeable battery (RB) has adequate energy storage capability but limited cycle lifetime. A super-capacitor (SC),

on the other hand, has lower energy storage capability but it can accommodate a potentially larger number of charging cycles.

2.2 Overview of some EH-WSN Routing Algorithms

In this section, we give an overview of some EH-WSN routing protocols and algorithms. The algorithms below can be roughly classified as not using clustering, using clustering, and routing in networks for special application environments. We note that although the presented algorithms differ in their system model, model assumptions, route selection methods, energy management, and performance measures, their design share a common principle which is aiming to maintain energy neutrality (e.g., balancing harvested and consumed energy) over certain time intervals in order to achieve good performance.

Table 2.2 highlights some features of the surveyed papers (we use letters C and J to denote conference and journal publications, respectively).

Table 2.2: Summary of selected energy harvesting aware routing protocols for EH-WSNs

Reference	**Methodology**	**Synopsis**
[24] (C 2010)	Simulation only	The paper presents an EH-WSN routing algorithm based on combining hop count distances and functions of node energies into "energy distance" measure used in distance vector routing framework.
[21] (C 2014)	Simulation only (using Matlab)	The paper proposes an EH-WSN routing algorithm based on modifying AODV routing algorithm for MANETs by replacing the hop count field in RREQ/RREP messages with "energy count" field.

[33] (J 2011)	Simulation only (using Matlab)	The paper presents a routing cost metric for EH-WSNs utilizing both rechargeable batteries (RBs) and super-capacitors (SCs). The design favours routes with more SC energy and harvesting rate.
[35] (C 2013)	Simulation only (using Omnet++)	The work presents an energy neutral routing algorithm that incorporates admission control of flows in a network that uses the Directed Diffusion paradigm [23].
[31] (J 2014)	Simulation only (using NS-2)	The authors formulate a route selection problem to minimize the amount of wasted harvested energy when some of the rechargeable batteries in a network become fully charged.
[34] (C 2013) [36] (J 2015)	Convex optimization + simulation (using MatLab)	The authors present an energy neutral clustering algorithm where cluster heads are assumed to be able to reach the sink node.
[27] (C 2015)	LP optimization + data analysis	The work discusses routing in WSNs embedded in exterior walls of buildings in Northern-climate areas and using harvested energy.
[6] (C 2015)	Simulation only (using Opnet)	The paper discusses the use of EH-WSNs in e-Health applications where traffic is categorized into compressed medical images, long data steams, and short data steams.

Routing Without Clustering: In [24], the authors present a distributed and adaptive routing algorithm for EH-WSNs with a single sink node. The algorithm aims at dynamically finding sustainable routes to the sink. A route is sustainable if it has sufficient energy to work for sometime. The work assumes that the stored energy in a node is measurable. To calculate sustainable routes, information about each available node energy (converted to a quantity called *local distance penalty*), and shortest hop count to sink (viewed as a

shortest structural distance) are exchanged between nodes. The local distance penalty is a real number derived from a node's stored energy such that low (respectively, high) node energy maps to high (respectively, low) real number. The algorithm adds hop counts to distance penalties to obtain a distance metric used in shortest path calculations. Shortest path calculations using the new distance metric proceeds as in Distance Vector (DV) routing. Events that trigger a node to send updates to neighbours include changes in node's local energy, and receiving updates from neighbours. Using simulation, the authors compare their algorithm, called DEHAR, with a simplified version of a routing algorithm utilizing the Directed Diffusion (DD) paradigm [23].

In [21], the authors modify the well-known Ad-hoc On-Demand Distance Vector (AODV) routing algorithm [38] for routing in Mobile Ad-hoc Networks (MANETs). The resulting energy harvesting-aware algorithm is denoted AODV-EHA. The prime modification is the use of an "energy count" quantity in place of "hop count" in the header of route request (RREQ) and route reply (RREP) messages. The energy count quantity is intended to be a prediction of the average transmission energy to successfully deliver a data packet from a source to a destination node. This average transmission energy takes into account the average number of transmission retries, the power consumed in processing and transmitting a packet, and the energy harvested during transmission. Using simulation, the authors compare their AODV-EHA algorithm with the AODV algorithm, and the DEHAR algorithm of [24].

In [33], the author present a routing cost metric for EH-WSNs where each node has a hybrid energy storage system (HESS) that combines a supercapacitor (SC), and a rechargeable battery (RB). The SC has low energy storage capacity but can accommodate frequent charge cycles. On the other hand, the RB has a higher energy storage capacity but limited charge cycles. The cost metric presented in [33] (discussed in more detail in Section A.1) assigns different weights to the residual energy in both units whilst favouring routes with more SC energy and harvesting rates. To prolong the battery lifetime, similar to the work in [26], the constructed cost metric assumes that the power management in each node directs the harvested energy to the RB when

its residual energy falls below a threshold value of the product of $(1 - D) \times$ the RB maximum capacity. Otherwise, the SC is charged. Here, D denotes the RB's designed *depth-of-discharge* (DoD), i.e., the maximum recommended fraction of battery capacity that can be withdrawn in one discharge cycle to maximize the battery's lifetime. The constructed cost metric is used to evaluate routes through different paths.

In [35], the authors present an energy neutral WSN routing algorithm (called ENR). The ENR algorithm is based on the Directed Diffusion (DD) paradigm [23]. We recall that the general DD paradigm allows a WSN to support many different task types. Each task description is named by a list of attribute-value pairs. For example, a vehicle-tracking task may use a type attribute for a mobile object to be detected, a duration for reporting matching events, and an interval specifying how frequently the matching events would be reported. A named task description constitutes an *interest* that is injected into the network from a sink node as a result of processing a query to the network. The sink initially and repeatedly diffuses an interest for *exploring* the possibility of obtaining matching replies. Subsequently, the sink may reinforce one particular neighbour in order to collect more data. The dissemination of an interest causes some participating nodes to set up a *gradient* (data rate and directions to forward events) at some participating nodes. The work in [35], implements a simplified version of the DD paradigm, adding an admission control step (to achieve energy neutrality) before a node can reinforce a flow that serves a particular interest. The admission control step utilizes a prediction of the harvested energy at each node, the duration and interval parameters of a new flow, and the contents of a node's interest cache that describes other flows the node serves.

In [31], the authors consider wastage of harvested energy when a node's finite-capacity battery reaches its limit (and the battery can not be overcharged). The authors formalize a route selection problem that aims at reducing network wide overcharging-wastage in each time slot of the network's operation. They present energy harvesting wastage-aware algorithm (called, EHWA) that selects a route by combining the *cost* associate with *energy con-*

sumption due to packet transmission and *energy wastage* due to finite-capacity battery being fully charged. Among other results, the authors show that a route that minimizes the overcharge-wastage is a route that maximizes the total network resultant energy at the end of each time slot. More mathematical details about this work is presented in Section A.7

Routing with Clustering: In [34, 36], the authors present an energy neutral clustering algorithm for EH-WSNs. The devised algorithm follows closely many concepts and ideas of the well-known LEACH (Low Energy Adaptive Clustering Hierarchy) algorithm [22]. We recall that LEACH divides time into rounds. Each round consists of two phases: a *set-up phase* where the network is partitioned into clusters, and a *steady-state phase* where each cluster head (CH) collects data from nodes in its cluster. LEACH assumes that each CH has enough power to reach the base station. LEACH estimates the optimal number of clusters to operate in each round, and uses a randomized process to select (on average) this optimal number of cluster heads. The energy neutral cluster (ENC) algorithm devised in [34, 36] also divides time into rounds (called time slots), and the algorithm partitions the network into a user-specified number of clusters (denoted K) during the first part of a time slot. The second part of each time slot is used to collect data in each cluster. Each round starts by selecting a cluster head group (CHG) that serves during the round. The size of the CHG is sufficient to ensure energy neutrality operation during a time slot. Scheduling the work of nodes in a CHG takes each node's energy into consideration. The authors formalize a convex optimization problem to find the optimal number of clusters. More mathematical details about this work are presented in Sections A.3 to A.6.

Routing for Special Application Environments: In [27], the authors consider the deployment of WSNs embedded in walls of buildings to collect information (such as humidity measurements sensed within walls) and send it to a sink node. Buildings in the work are assumed to exist in Northern climate areas where during colder months the temperature difference between indoors and outdoors provides opportunity to generate electricity from thermal effects. Temperature data collected over two years is used to estimate

the amount of energy generated every period (e.g., day or week) during each year. The work formulates two routing problems where in each problem traffic generated by each node is routed to a sink node. The formulation assumes the use of splittable flows (where several non-disjoint paths can serve the traffic stream generated by each node). Each routing problem is formulated by giving a corresponding linear program (LP). The first routing problem aims at maximizing the total flow (from all nodes) to the sink node. The formulation considers the available energy at each node. To achieve fairness in serving all nodes, the second routing problem aims at maximizing a single flow value that can be supported between each node in the network and the sink node. The default way of using the constructed LPs is to repeatedly solve the optimization problem(s) by a central node (e.g., the sink node), and then inform other nodes with routing decisions. In [27], the authors discuss a table driven method (called a seasonal routing algorithm) where routing can be done by "seasonal" adjustments done by each node independently without requiring solving an optimization problem at each step during the lifetime of the network.

In [6], the authors discuss "Smart Energy Harvesting Routing" protocol (called, SEHR) for data handling in WSN based e-Health systems (WSNEH). Application data is categorized into three types:

- Compressed high resolution medical images (each image is about 6 Mbits)
- Big sized data (configured to 4.5 Mbits per stream), and
- Regular sized data (configured to 2 Kbits per stream)

Each data type has additional requirements on delivery time and route stability (however, such aspects are not well detailed in the paper). The authors devise a route selection algorithm where the destination node of a flow evaluates a function (called the Q factor) for each possible route. The types of traffic flows passing through each node on a route, and the energy state of the node are considered by the function. The authors experiment with both RF energy harvesting, and solar energy harvesting.

2.3 Overview of some Analytical Models

Please refer to Appendix A for the material in this section (if desired).

2.4 Concluding Remarks

In this chapter, we have briefly discussed energy harvesting sources, and energy harvesting architectures used in EH-WSNs. We have also provided an overview of some recent EH-WSN routing algorithms and we ended the chapter with some mathematical analysis for selected routing algorithms. In the next chapter, we formalize a path exposure reliability problem where nodes utilize energy replenishment.

Chapter 3

Path Exposure with Range Uncertainty

Many WSNs deployed in outdoor areas are intended to detect application specific events. Such outdoor networks are well suited to operate on energy harvested from the ambient environment. In surveillance applications, detecting a traversal across a geographic area guarded by a WSN is one of the early studied WSN applications. In particular, early work in this direction has considered the notion of *path exposure*. As mentioned in Section 1.4, path 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.

In this chapter, we consider EH-WSNs, where fluctuations in a node's energy levels is assumed to cause the node to change its transmission range. In Section 3.1, we formalize a new path exposure reliability problem called *path exposure with range uncertainty*, denoted EXPO-RU, where we want to quantify the ability of an EH-WSN to jointly detect and report a traversal along a given path across an EH-WSN. Then, we show that the EXPO-RU is #P-hard. In Section 3.2, we formulate an *extension to pathset* problem, denoted E2P, for the EXPO-RU problem. We propose an efficient heuristic algorithm to solve the E2P problem, and the devised algorithm is used to design both *iteratively* and *non-iteratively* improvable LB algorithms for the EXPO-RU problem.

Some of the results in this chapter appear in [8].

3.1 Problem Formulation

In this section, we present the network model used to formalize the EXPO-RU problem, define the EXPO-RU problem, and show that the problem is #P-hard.

3.1.1 Network Model

During an interval of time of interest, a node x in an energy replenishment WSN can operate at different power levels. We adopt a model that uses in its basic form a simple 3-state node model where the analysis designates one range of power levels as a *full* power range, and a second disjoint range as being a *reduced* power range. A node lacking power in these ranges is considered *failed*. This 3-state node model, and our devised algorithms, can be extended to deal with a more fine-grained model that employs more power states at the expense of requiring possibly higher algorithmic time and space complexities.

For any node x, the above events occur with probabilities denoted $p_{full}(x)$, $p_{red}(x)$, and $p_{fail}(x)$ ($p_{fail}(x) = 1 - p_{full}(x) - p_{red}(x)$). Loss of a node's power affects its communication, sensing, and computing abilities. Since wireless transmission is the most power demanding activity in many WSNs, we assume that a node operating in a reduced power state incurs a reduction of its transmission range. Different nodes in the network may have different state probability distributions, and different transmission ranges when they operate in these states.

The overall WSN is modelled by a probabilistic directed graph $G = (V \cup \{s\}, E)$ where s is a distinguished sink node, and E is a set of arcs (directed edges). We assume that s is not subject to power fluctuations, and it does not perform sensing. A non-sink node x can be in a state $s_x \in \{full, reduced, fail\}$, during an interval of time when the network is analyzed.

Estimating Node State Probabilities: One way to estimate the needed node state probabilities $p_s(x)$ for node x and state s is to use Monte Carlo simulation. We divide time into fixed length slots, and simulate the network

under a specified traffic load for a sufficiently long time. We then take the fraction of the number of time slots when x is in state s over the total number of simulation slots as an estimate of $p_s(x)$.

3.1.2 Problem Definition

As mentioned in Section 1.4, the path exposure problem is well studied in the literature (see, e.g., [15, 32]). The problem asks for computing the probability of detecting a moving target on a given path over a period of time through a WSN field. Let $\mathbf{P}$ be an intrusion path across the WSN that we need to monitor for unauthorized intrusion (see, e.g., Figure 3.2.1). Assume the nodes that can sense the path $\mathbf{P}$ (called *sensing nodes*) are known from the geometric layout of the network. Also, assume that in order to detect an intrusion event, we need at least k_{req} sensing nodes to be able to reach the sink and report the detection successfully. Therefore, our problem can be defined as follows.

> **Definition (the EXPO-RU Problem)**: Given a WSN that is modelled by a probabilistic graph $G = (V \bigcup \{s\}, E)$ where s is the sink node, an integer $k_{req} \geq 1$, a possible intrusion path $\mathbf{P}$ where nodes that can sense $\mathbf{P}$ are known, and each node $x \in V$ works either in a full or reduced power states with probabilities $p_{full}(x)$ and $p_{red}(x)$, respectively, where $p_{full}(x) + p_{red}(x) \leq 1$, compute the probability $\text{Expo}(G, p, \mathbf{P}, k_{req})$ that $\mathbf{P}$ can be jointly detected by at least k_{req} sensing nodes that can reach the sink. ■

To simplify notation, when no confusion arises, we sometimes abbreviate $\text{Expo}(G, p, \mathbf{P}, k_{req})$ as $\text{Expo}(G, p, k_{req})$ or $\text{Expo}(G, k_{req})$.

3.1.3 Complexity Analysis

Theorem 3.1.1 below shows that the EXPO-RU problem is #P-hard. The class #P-hard (or, #P-complete) problems captures the computational complexity of counting problems. In [20], the authors tackle a different version of the path exposure problem, called the EXPO problem, defined on conventional WSNs, in which each node x can successfully forward packets to its neighbours

with probability $p_{relay}(x)$, and sense a target with probability $p_{sense}(x)$. The authors, in [20], show that a restricted version of the EXPO problem is #P-hard. In the restricted version of the EXPO problem the input is a partial grid network, only one node x has a non-zero probability of sensing the intruder path $\mathbf{P}$, and $k_{req} = 1$. Theorem 3.1.1 below uses this restricted version of the EXPO problem to show the complexity of the EXPO-RU problem.

Theorem 3.1.1 EXPO-RU *is #P-hard.*

Proof. We reduce the restricted version of the EXPO problem to the EXPO-RU problem in polynomial time as follows. Let $(G, \mathbf{P}, k_{req}, p_{relay}(.), p_{sense}(.))$ be an instance of the restricted EXPO problem (presented in [20]) where G is a grid network, $\mathbf{P}$ is an intrusion path that can be monitored by one node x where $p_{sense}(x) \neq 0$ (other nodes have $p_{sense}(x) = 0$).

We transform in polynomial time the above restricted instance of EXPO problem to an instance of our EXPO-RU problem. The transformation creates an instance $(G', \mathbf{P}', k'_{req}, p_{full}(x), p_{red}(x))$ of the EXPO-RU problem where G', $\mathbf{P}'$, and k'_{req} correspond to G, $\mathbf{P}$, and k_{req} in the above instance of the EXPO problem, respectively. In the EXPO-RU problem, set $p_{full}(x) = 0$, and $p_{red}(x) = p_{relay}(x)$ for each node x. This mapping from the restricted EXPO problem to our EXPO-RU problem preserves the value of the solution and hence proves the theorem. ■

3.2 The E2P Problem for EXPO-RU

In this section, we present concepts needed to design our LB algorithms for the EXPO-RU problem, and formalize a pathset extensibility problem, called the *Extension to Pathset* (E2P) problem. Then, we present a heuristic algorithm to solve the E2P problem.

3.2.1 Concepts Needed for Algorithms

Network States. When each node in V is assigned one of its possible states, we obtain a *network state* of G (a total of $3^{|V|}$ states exist). Using (x, s_x) to

denote the event that node x is in state s_x (with probability, denoted $p_{s_x}(x)$ or $p(x, s_x)$), a network wide state S is specified by a set $S = \{(x, s_x) : x \in V, s_x \in \{full, reduced, fail\}\}$ of node-state pairs, where $|S| = |V|$.

We assume that nodes can be in different states independently of each other. Thus, network state S arises with probability $\Pr(S) = \prod_{(x,s_x)\in S} p(x, s_x)$.

Network Configurations. A network *configuration* C is a obtained by assigning a state to each node in a subset of V. Using the above set notation to describe C, we then have $|C| \leq |V|$ (an empty set is a valid network configuration).

Pathset. A *pathset* is an operating network configuration that has at least k_{req} sensing nodes that can reach the sink and monitor the given intrusion path **P**. A *minpath* is a minimal pathset.

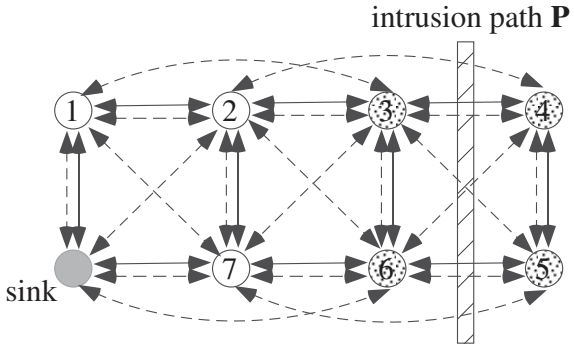


Figure 3.2.1: An instance of the EXPO-RU problem

Example 3.2.1 Fig. 3.2.1 illustrates an instance of the EXPO-RU problem where dashed (solid) lines represent reachability by full (respectively, reduced) transmissions. To avoid cluttering the diagram, two arcs (x, y) and (y, x) are drawn as a double-arrowed line connecting x and y. In the example, we assume that if node x can reach node y while x in some state (either $full$ or $reduced$) then y can also reach x while y is in the same state. Nodes that can sense **P** appear as dotted circles. $C = \{(4, full), (6, reduced)\}$ is a configuration that leaves the remaining five non-sink nodes *free*. For $k_{req} = 2$, C is extensible to a pathset such as $S = \{(4, full), (6, reduced), (7, reduced)\}$. ■

An Exact Algorithm. Given an instance of the EXPO-RU problem, an exact algorithm may proceed by generating the set $\mathcal{OP}(G, k_{req})$ of all pathsets

in G, and outputing

$$\text{Expo}(G, p, k_{req}) = \sum(\Pr(S) : S \in \mathcal{OP}(G, k_{req})).$$

The worst case running time of such an exact algorithm, however, is $O(3^{|V|})$ time.

Our work here devises polynomial time algorithms to compute lower bounds (LBs) on the exact solutions of EXPO-RU problem.

Here, we define a pathset extensibility problem, called the Extension to Pathset (E2P) problem as follows.

> **Definition (the E2P Problem)**: Given an instance $(G, p, \mathbf{P}, k_{req})$ of the EXPO-RU problem, and an input configuration $C = \{(x, s_x) : x \in V, s_x \in \{full, reduced, fail\}\}$ of node-state pairs (C can be empty), find a configuration C_{new} (composed of full and reduced nodes) that extends C to a valid pathset (if possible) such that: (**a**) $C \bigcap C_{new} = \phi$, and (**b**) C_{new} has the highest occurrence probability. ■

The next Section discusses an efficient heuristic algorithm to solve the E2P problem.

3.2.2 The E2P Algorithm

3.2.2.1 Algorithm Idea

Our main contribution in this section is a heuristic algorithm, called E2P, to solve the E2P problem. The algorithm has two main ideas. Firstly, we employ a transformation of the input probabilistic graph G to an (ordinary) weighted directed graph G' that allows us to find optimized structures using conventional single-destination shortest paths algorithms [17]. In particular, for each non-sink node x in G, we associate a weight $w(x, y)$ with each arc (x, y) where x can reach y while x in some state $s_x \in \{full, red\}$ so that $w(x, y)$ is small when the probability $p_s(x)$ is large (and vice versa). Arc weights in G' can be added (as with distances), so as to allow using shortest paths algorithms.

Secondly, the overall algorithm performs a number of iterations. Each iteration aims at extending the computed extension C_{new} (by adding more node-state pairs) so that we increase the number of nodes that can jointly sense the intrusion path $\mathbf{P}$ and report the intrusion events to the sink.

To explain the first main idea, we introduce the following definition. The definition uses the input probabilistic graph (G, p), an input configuration C (C can be empty), and a currently computed extension configuration C_{new} (C_{new} starts empty and changes as more iterations are done).

Definition: The graph $G' = \mathit{logp\_dist}(G, C \cup C_{new})$ is a weighted directed graph obtained from G, C, and C_{new} as follows:

1. If a node x is assigned the failed sate in configuration $C \cup C_{new}$ then delete x from G'.

2. Else, if x is assigned the full (or, reduced) state in $C \cup C_{new}$ then G' has an arc (x, y) for each node y reachable from x in this state. We set $w(x, y) = 0$.

3. Else, if x, $x \neq \mathit{sink}$, is a free node in $C \cup C_{new}$ (i.e., $x \notin (C \cup C_{new})$ where $p_{full}(x) \neq 0$ (or, $p_{red}(x) \neq 0$) then G' has an arc (x, y) for each node y reachable from x in this state. We set $w(x, y) = -\log(p_{full}(x))$ (respectively, $w(x, y) = -\log(p_{red}(x))$).

Thus, all arc weights in G' take on finite non-negative values. ■

To explain the second main idea, we denote by $k(C \cup C_{new})$ the number of operating nodes in the configuration $C \cup C_{new}$ that can sense the intrusion path $\mathbf{P}$, and report the intrusion to the sink. Thus, a configuration $C \cup C_{new}$ is a pathset if $k(C \cup C_{new}) \geq k_{req}$. The algorithm starts each iteration with the input configuration C, and a currently computed extension C_{new}. If $(C \cup C_{new})$ is a pathset the algorithm terminates successfully. Else, the iteration tries to compute an extension C'_{new} of $C \cup C_{new}$ such that $k(C \cup C_{new} \cup C'_{new}) > k(C \cup C_{new})$.

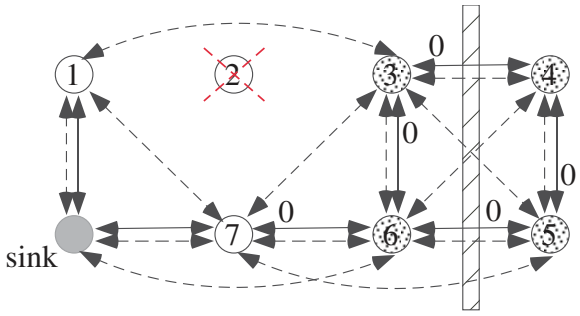


Figure 3.2.2: Construction of G'

Example 3.2.2 Fig. 3.2.2 illustrates the graph $G' = logp\_dist(G, C \cup C_{new})$ where G is the graph in Fig 3.2.1, $C = \{(2, fail), (4, reduced), (6, reduced)\}$, and $C_{new} = \emptyset$. In G' we have the following:

1. Node 2 is deleted since $(2, fail) \in C$
2. The weight of each reduced range arc out of nodes 4 and 6 is set to zero since $\{(4, reduced), (6, reduced)\} \subseteq C$
3. The weight of each of the remaining arcs is obtained using rule 3 above.

■

3.2.2.2 Algorithm Details

The main steps performed by function E2P in Fig. 3.2.3 proceed as follow. Step 1 initializes the sought after solution C_{new} to empty. Step 2 performs a number of iterations until $k(C \cup C_{new}) \geq k_{req}$. Step 3 transforms the probabilistic graph (G, p), and the current configuration $C \cup C_{new}$ to a directed distance graph. Step 4 utilizes a single-destination shortest paths algorithm to find (if possible) a minimum weight path $P = (x_0, x_1, x_2, \cdots, sink)$ from a sensing node x_0 that does not reach the sink in the current configuration $C \cup C_{new}$ to the sink. Else, Step 5 returns with failure (-1). Step 6 extracts from the computed path P nodes that are not currently in $C \cup C_{new}$. The step then assigns to each such a node x_i a state (either full, or reduced) depending on the range of the arc $(x_i, x_{i+1}) \in P$.

Function E2P($G, p, C, C_{new}, k_{req}$)
Input: An instance (G, p, C, k_{req}) of the $E2P$ problem

Output: If the $E2P$ problem has a solution return $+1$, and a solution in C_{new}. Else, return -1.

1. Set $C_{new} = \emptyset$
2. **while** $(k(C \cup C_{new}) < k_{req})$
   {
3. Set $G' = logp\_dist(G, C \cup C_{new})$
4. In G', find a minimum weight directed path $P = (x_0, x_1, \cdots, sink)$ from a node x_0 that can sense the intrusion (but can not currently reach the sink) to the sink
5. **if** (P does not exist) return -1
6. **Else,** form configuration C'_{new} from the found path P that includes only nodes not currently in $C \cup C_{new}$
7. Add C'_{new} to C_{new}.
   }
8. **return** $+1$

Figure 3.2.3: Function E2P for the EXPO-RU problem

3.2.2.3 Correctness and Running Time

Theorem 3.2.1 Let (G, p, C, k_{req}) be an instance of the EXPO-RU problem where C is a configuration of G that can be extended to a pathset then

1. The above algorithm finds a solution C_{new}, and
2. At each iteration, the computed configuration C'_{new} is optimal for the iteration

Proof. Part 1 follows since the algorithm finds a collection of paths from the sensing nodes in G to the sink. Adding the nodes of any such a path to the currently computed solution C_{new} does not prohibit the identification of other paths in subsequent iterations.

Part 2 follows since in each iteration the weight $w(P)$ of a computed path is minimum possible, since we utilize a single-destination shortest paths algorithm. In addition, the transformation guarantees that a minimum weight path in G' maps to an incremental configuration C'_{new} with the highest possible $\Pr(C'_{new})$. ■

Running time. Function E2P runs in time $O(k_{req} \cdot (n+m))$ on a probabilistic graph G with n nodes and m arcs. This follows since each iteration of function E2P constructs a directed graph G', runs a single-destination shortest paths algorithm, and if a path P is found, it extracts the new nodes and updates the computed solution C_{new}. Running a shortest paths algorithm requires $\Theta(n+m)$ time (or better) (see, e.g., [17]). Performing the other steps in each iteration can be done in $O(n+m)$ time.

3.3 Algorithms for Lower Bounds

In this section we highlight ideas of obtaining iteratively and non-iteratively improvable algorithms to compute LBs on exact solutions of the EXPO-RU problem. Both algorithms rely on using function E2P described above.

3.3.1 An Iteratively Improvable Algorithm

As mentioned in 1.5, an iteratively improvable LB algorithm produces an exact solution if allowed to run to completion. Else, it outputs a LB on the solution. Iteratively improvable algorithms typically rely on methods for systematically generating a maximal set of pathsets that suffices to compute the exact solution. As mentioned in Section 1.2.3, the *factoring* method, discussed in [7], is an algorithmic framework that serves the above purpose; the method applies to a broad class of network reliability problems. In [20], the authors adapt the factoring method to compute LBs and UBs on a variant of the path exposure problem where a node may fail in communication and/or sensing.

The factoring algorithm presented in [20] computes a LB by systematically generating a set of pathsets $\{P_1, P_2, \cdots, P_r\}$ such that

$$\text{Expo}(G, p, k_{req}) \geq \sum_{i=1}^{r} Pr(P_i) \tag{3.3.1}$$

One way to ensure that the above equation holds is to ensure that any pair of generated pathsets P_i and P_j have at least one common node, say x, such that P_i and P_j assign two different states to node x. We call such pathsets (or

configurations) *s-disjoint*. For the EXPO-RU problem, the two configurations $C_1 = \{(1, reduced), (2, fail)\}$, and $C_2 = \{(1, full), (2, fail)\}$ are s-disjoint since node 1 is assigned two different states in C_1 and C_2.

In more detail, the factoring algorithm described in [20] iteratively generates s-disjoint configurations in a tree-like manner, starting from the empty configuration as the root of the tree. At each iteration, the algorithm selects a configuration C that is perceived to be most promising (in terms of its contribution to the computed solution). Configuration C is then used to generate new pairwise s-disjoint configurations that are added to the tree. Upon generating a configuration, the algorithm calls a suitable E2P function to decide on its extensibility to a pathset. By deciding on the extensibility of a configuration immediately after being generated, the factoring algorithm limits the number of configurations considered to be active in each iteration.

Section 3.4 presents results obtained by integrating function E2P within the factoring framework, and we refer to the obtained LB in this way as the factoring bound.

3.3.2 A Non-iteratively Improvable Algorithm

As mentioned in Section 1.5, a non-iteratively improvable LB algorithm computes a single value after processing a given problem instance. Effective non-iteratively improvable algorithms are useful both as standalone tools, and also as subroutines called from within an iteratively improvable algorithm to handle some cases.

One such algorithm computes a LB from a given set of pathsets $\{P_1, P_2, \cdots P_r\}$ that are pairwise **node-disjoint**. Computing lower bounds from node-disjoint pathsets has been extensively studied in the context of many network reliability problems where each node can either be *operating* or *failed* (see. e.g., [7]). In Section 1.2.3, we presented a well-know formula for obtaining a LB from edge-disjoint pathsets for the Rel_2 problems where each element (e.g., edge) can be either operating or failed.

In this section, we discuss using a similar formula to obtain a LB for the 3-state EXPO-RU problem as follows.

1. Compute a set $\{p_1, p_2, ..., p_r\}$ of node-disjoint pathsets

2. For each node x where $(x, reduced)$ appears in a pathset p_i, set the occurrence probability $p_{occ}(x) = p_{red}(x) + p_{full}(x)$

3. For each node x where $(x, full)$ appears in a pathset p_i, set the occurrence probability $p_{occ}(x) = p_{full}(x)$

4. For each pathset $p_i = \{(x, s_x) : \text{node } x \text{ is either full or reduced}\}$, set $\Pr(p_i) = \prod_{(x,s_x)\in p_i} p_{occ}(x)$

5. Compute a LB on the solution using formula (3.3.2):

$$\Pr[\text{at least one pathset } p_i \text{ occurs}] = 1 - \prod_{i=1}^{r}(1 - \Pr(p_i)) \qquad (3.3.2)$$

We note that the *E2P* algorithm discussed in Section 3.2 (which gives a good pathset extension for an input graph G and a configuration C) can be used repeatedly to obtain a good set $\{p_1, p_2, ..., p_r\}$ of node-disjoint pathsets. We refer to the obtained LB in this way as the NDP bound.

In the next section, we conduct experiments to compare the strength of LBs obtained by the NDP method with the LB obtained by executing a fixed number N ($N = 1000$) of factoring iterations.

3.4 Numerical Results

In this section, we present some numerical performance results of our devised algorithms. We use as test networks, the class of double-diagonal grids (called x-grids, for short) of dimensions $W \times W$, $W \geq 2$, where the rows (or, columns) are numbered $0, 1, \cdots W-1$. The grid topology is one of the commonly referenced topologies in WSN deployments. Their structural regularity simplifies analyzing the numerical findings. Fig. 3.4.1 illustrates a 3×3 x-grid where the sink is located at (x, y)-coordinate $(0, 0)$, and the intrusion path $\mathbf{P}$ is placed vertically between the leftmost 2 columns. Solid (dashed) edges represent reduced (respectively, full) transmission ranges. For simplicity, we assume that nodes have identical transmission ranges when operating in their reduced (or

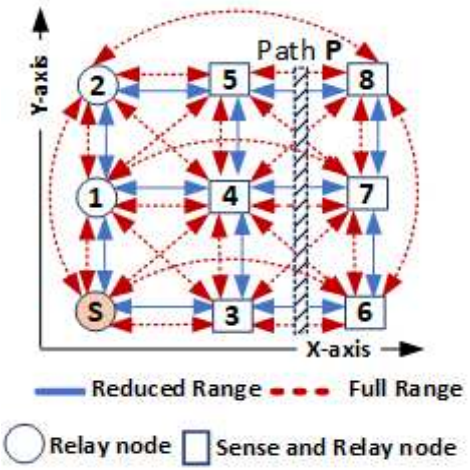


Figure 3.4.1: A 3×3 x-grid network with an intrusion path **P**

full) power states. In the experiments, only nodes placed on the left and right of **P** can sense the path.

Graph legend. Label **NDP** refers to LBs from node-disjoint pathsets, and the **Factoring** label refers to LBs from the factoring algorithm.

3.4.1 Work Done by the Factoring Algorithm

The factoring algorithm discussed in Section 3.3 generates a maximal set of pairwise s-disjoint pathsets when it runs to completion. This enables the algorithm to compute the exact Expo(G, p, k_{req}). The algorithm avoids the generation of all possible network states by discarding (at an early stage) configurations that can not be extended to pathsets. Table 4.1 lists the obtained results when solving the EXPO-RU **exactly** on x-grids with different sizes, and k_{req} values. Each network is processed within one minute of user time. As can be seen, the factoring algorithm finds the exact solution by examining only a small fraction of the maximum number of possible network states.

3.4.2 Experiments with Varying the Network Size and k_{req}

An important aspect in the design of a WSN is for the overall WSN to achieve performance higher than that achieved by its individual nodes. This aspect

Table 3.1: Exact computations by the factoring algorithm

x-grid size	k_{req}	**Configurations generated**	**Pathsets found**	**Maximum possible number of states**
2×2	1	7	6	3^3
3×3	1	15	14	3^8
4×4	1	645	530	3^{15}
2×2	2	17	12	3^3
3×3	2	67	56	3^8
4×4	2	357	274	3^{15}
2×2	3	15	8	3^3
3×3	3	207	158	3^8
4×4	3	675	484	3^{15}

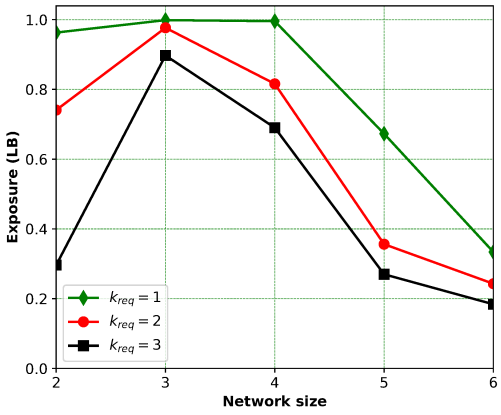
is expected as a consequence of the cooperative work of all sensors in the network. In this set of experiments, we vary the size of a $W \times W$ x-grid where $W \in [2, 6]$, and vary k_{req} in the range $[1, 3]$. For each non-sink node x, we set $p_{full}(x) = p_{red}(x) = p_{fail}(x) = \frac{1}{3}$. Fig. 3.4.2a presents the obtained results using the factoring algorithm after performing 1000 iterations.

In all networks, the sink is assumed not to sense the intrusion path, thus the 2×2 network has only 3 sensing nodes, whereas, the 3×3 network has 6 sensing nodes. The shown values for x-grids with $W = 2$ to 4 are exact solutions. For x-grids of widths $W = 5$ and 6, and $k_{req} \geq 2$, the computed LB indicates that the achieved exposure is no more than the probability that a node operates either in the full or reduced states. These preliminary results suggest that tuning the performance of networks relying on energy replenishment requires careful investigations before deployment.

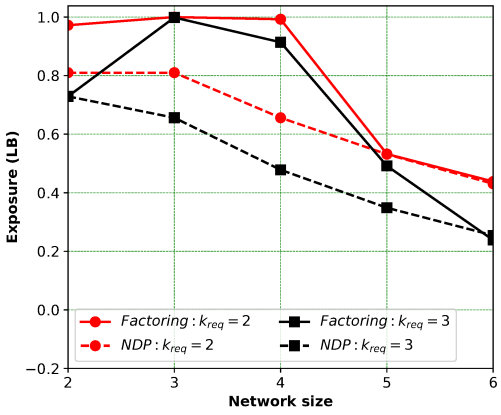
3.4.3 Comparison Among Lower Bounds

Non-iteratively improvable LB algorithms like the NDP algorithm are useful as a standalone tools as well as part of an iteratively improvable framework. In this section, we evaluate the strength of the NDP algorithm versus executing a fixed number N of iterations of the factoring algorithm (we use $N = 1000$).

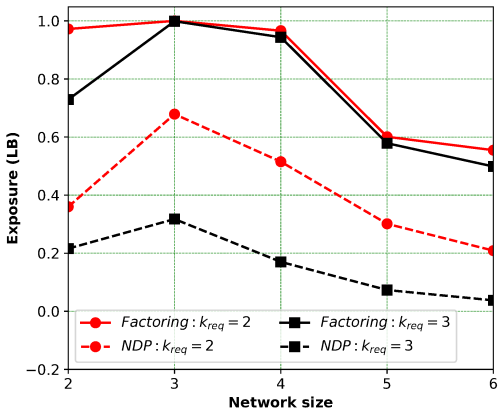
In our implementation of the factoring algorithm, performing N iterations



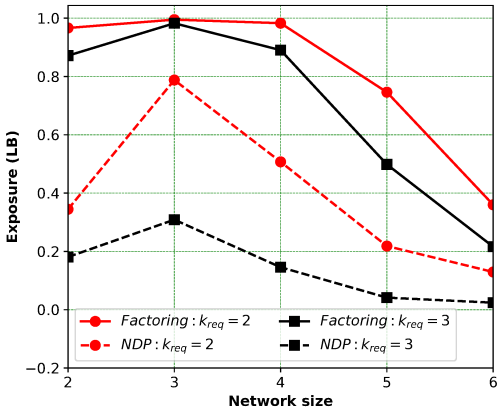
(a) Exposure versus grid width W (varying k_{req})



(b) Exposure versus grid width W ($p_{full} = 0.3$, $p_{red} = 0.6$)



(c) Exposure versus grid width W ($p_{full} = 0.6$, $p_{red} = 0.3$)



(d) Exposure versus grid width W (random state probabilities)

Figure 3.4.2: The obtained numerical results

requires making more than N calls to function E2P to check in each iteration the extensibility of all configurations derived from the particular configuration selected for processing in the iteration.

In comparison, in our experiments, the NDP algorithm generates at most 3 node-disjoint pathsets for any x-grid network used in the plots.

Fig. 3.4.2b and Fig. 3.4.2c illustrate the obtained results using the (p_{full}, p_{red}, p_{fail}) distributions of $(0.3, 0.6, 0.1)$, and $(0.6, 0.3, 0.1)$, respectively. Fig. 3.4.2d presents the results of using random state probabilities

We remark that the NDP algorithm, despite its much lower complexity, is sometimes able to compete (or, even outperform) the 1000 iterations performed

by the factoring algorithm. Although this finding may not generalize to a larger network, the obtained results encourage the use of the NDP algorithm as part of a factoring framework to compute LBs on some configurations.

3.5 Concluding Remarks

In this chapter, we consider EH-WSNs, where fluctuations in a node's energy levels is assumed to cause the node to change its transmission range. In this context, we formalize the EXPO-RU problem that seeks to quantify the ability of an EH-WSN to jointly detect and report a traversal along a given path across an EH-WSN to a sink node. We show that the EXPO-RU problem is #P-hard, and we design both iteratively and non-iteratively improvable algorithms for the EXPO-RU problem to derive lower bounds on the network's ability to detect intrusion along a given path across an EH-WSN monitored field.

Chapter 4

The E2C Problem for the EXPO-RU Problem

In this chapter, we extend our work in Chapter 3 by designing UB algorithms for the EXPO-RU problem. In Section 4.1, we review the system model as well as key definitions and assumptions for the EXPO-RU problem. In Section 4.2, we formulate an *extension to cutset* problem, denoted E2C, for the EXPO-RU problem. Then, we propose two efficient heuristic algorithms to solve the E2C problem. The devised algorithms are used to design both iteratively and non-iteratively improvable UB algorithms for the EXPO-RU problem.

Some of the results in this chapter appear in [10, 11].

4.1 System Model

In this section, we briefly review the used network model for the EXPO-RU problem as well as important definitions and assumptions.

4.1.1 Network Model

We adopt a network model similar to the one used in Chapter 3. The following summarizes the main assumptions. We consider EH-WSNs that rely on energy harvesting (e.g., solar energy). It is assumed that the system time is slotted into a set of time slots denoted $t_1, t_2, .., t_k$ where the energy harvesting rate of each non-sink node may be different during each time slot. The sink node is assumed to have unlimited energy. Fluctuations in a node's energy level affect its communication, sensing and processing. However, since wireless transmission is the most power consuming activity in many EH-WSNs, we assume that a reduction in a node's available energy affects primarily its transmission range. It is also assumed that there is an energy management unit (EMU) that controls states of each node based on its available stored energy level, and it controls the node's transmission ranges. So, we model the fluctuations in a node's stored energy by associating a probability that a node works either in full power, reduced power, or node can not work at all when its energy is depleted.

In particular, we adopt a model that uses in its basic form the following 3-state node model for each non-sink node x in the network:

- If x's stored energy level lies within a designer specified range (e.g., say $[70\% - 100\%]$), x is assumed to be able to communicate in full transmission power; then x is considered to be in the $full$ energy state (with probability $p_{full}(x)$).

- Else, if x's stored energy level lies within a lower range (e.g., say $[30\% - 70\%]$), x is assumed to be communicating with reduced transmission power; then x is considered to be in the *reduced* energy state (with probability $p_{red}(x)$).

- Else, x has no enough energy to communicate, then x is in the $fail$ state with failure probability $p_{fail}(x) = 1 - (p_{full}(x) + p_{red}(x))$.

The overall EH-WSN is modelled by a probabilistic directed graph $G = (V \bigcup \{s\}, E)$ where V is a set of nodes in G, s is a distinguished sink node, and E is a set of directed edges. It is assumed that the sink node s does not perform sensing tasks and it is not subject to power fluctuations. For any non-sink node $x \in G$, x can be in a state $s_x \in \{full, reduced, fail\}$ during a period of time when the network G is analyzed.

4.1.2 Review of the EXPO-RU Problem

As mentioned in Chapter 3, the EXPO-RU problem uses a probabilistic graph (G, p) to model the network and the problem formulation assumes that different nodes behave independent of each other. During a short random time interval, the network G is in some **network state** S where each node x is in some state $s_x \in \{full, reduced, fail\}$. We use $S = \{(x, s_x) : x \in V, s_x \in \{full, reduced, fail\}\}$ to refer to any such network state. The probability that a given network state S arises is $\Pr(S) = \prod_{(x,s_x)\in S} p(x, s_x)$. Each network state S is either *operating* or *failed*. To form an operating network state S, node states in S should be such that at least k_{req} sensing nodes in S can reach the sink node. Else, S is a *failed* state. The EXPO-RU problem is to compute the probability that the network G is some operating state. We denote such probability by $\text{Expo}(G, p, \mathbf{P}, k_{req})$, or $\text{Expo}(G, p)$ for short.

4.2 The E2C Problem for EXPO-RU

In this section, we present concepts needed to design our UB algorithms for the EXPO-RU problem, and formalize a cutset extensibility problem, called the *Extension to Cutset* (E2C) problem. Then, we present two heuristic algorithm to solve the E2C problem, and use the devised algorithms to design both *iteratively* and *non-iteratively* improvable UB algorithms for the EXPO-RU problem.

4.2.1 Concepts Needed for Algorithms

For every non-sink node $x \in G$, x can be in a state $s_x \in \{full, reduced, fail\}$ at any interval of time with probabilities $p_{full}(x)$, $p_{red}(x)$, and $p_{fail}(x)$, respectively, where $p_{fail}(x) = 1 - (p_{full}(x) + p_{red}(x))$.

We need the following concepts defined in Chapter 3. If all nodes in G are assigned possible states, we get a **network state** S of the graph G where $S = \{(x, s_x) : x \in V, s_x \in \{full, reduced, fail\}\}$, where $|S| = |V|$. On the other hand, if some nodes in V (but not necessary all nodes) are assigned possible states, we obtain a **network configuration** C (C can be the empty set) where $|C| \leq |V|$. A node x that is not assigned a state in C is called a $free$ node in C. A configuration C that has k_{req} sensing nodes and guarantees detection and reporting of unauthorized intrusion across the path **P** is called a **pathset**.

In this chapter, we also need the following definition. If a configuration C leads to a failed state of the whole network even if all free nodes in C are assigned the $full$ state, then C is called a **cutset**.

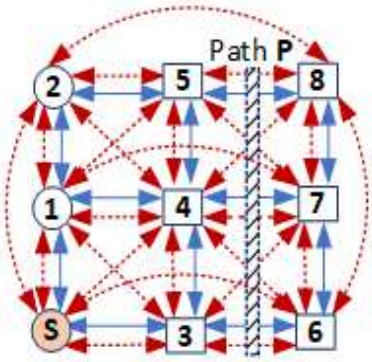


Figure 4.2.1: An instance of the EXPO-RU problem

Example 4.2.1 Fig. 4.2.1 shows an instance of the EXPO-RU problem for a 3×3 double-diagonal grid network G where the dashed lines (coloured red) represent communications in full power whereas the solid lines (coloured blue) represent communications in reduced power. In Fig. 4.2.1, arcs (x, y) and (y, x) are drawn as a double-arrowed line. In this example, we assume that if node x reaches node y while x is in a state (either $full$ or $reduced$) then y

can also reach x while y is in the same state. In Fig. 4.2.1, squares represent nodes that can sense the intrusion path $\mathbf{P}$. Let $C = \{(3, fail), (6, full)\}$ be a given network configuration and let $k_{req} = 2$. Then, C can be extended to a pathset by adding nodes $1, 4$ in the *reduced* states where $S = \{(3, fail), (6, full), (1, reduced), (4, reduced)\}$. Here, nodes 4 and 6 can sense the intrusion and reach the sink. However, if $C = \{(1, fail), (2, reduced), (3, fail), (4, reduced), (6, reduced)\}$, then C is a cutset *even if all remaining nodes in* G *are assigned the* $full$ *state since* C *isolates the sink* s. ■

Now, we define a cutset extensibility problem, called the Extension to Cutset (E2C) problem as follows.

> **Definition (the E2C Problem)**: Given an instance $(G, p, \mathbf{P}, k_{req})$ of the EXPO-RU problem, and an input configuration C that is not a pathset where $C = \{(x, s_x) : x \in V, s_x \in \{full, reduced, fail\}\}$ of node-state pairs (C can be empty), find a configuration C_{new} (composed of reduced and failed nodes) that extends C to a valid cutset (if possible) such that: (**a**) $C \bigcap C_{new} = \phi$, and (**b**) C_{new} has the highest occurrence probability. ■

The next sections presents two heuristic algorithms to solve the E2C problem by utilizing two well-know techniques:

- The *E2C-BFS* algorithm that utilizes breadth-first-search (BFS) technique (cf. Section 4.2.2).
- The *E2C-MaxFlow* algorithm that utilizes a solver to the max-flow problem (cf. Section 4.2.3).

4.2.2 The E2C-BFS Algorithm

Our main contribution in this section is a heuristic algorithm that utilizes breadth-first-search (BFS) [17] to solve the E2C problem, denoted E2C-BFS. Given a graph G on $V \bigcup \{s\}$ nodes, the *BFS* layering of G, using the sink node s as a root, is a partition of the set nodes V into disjoint subsets (called, layers) denoted $L_0, L_1, L_2, ..., L_r$ where layer $L_0 = \{s\}$. In a BFS tree directed

towards the sink, a node x belongs to layer L_i, $i \geq 0$, if and only if the shortest directed path from x to s has exactly i arcs. For many graphs, the sink node s has a few neighbours. So, layer L_1 is typically small relative to other layers. Hence, L_1 forms a good start point to compute a cutset.

For the EXPO-RU problem, we note that depending on the input configuration C, nodes in layer L_1 alone may not be sufficient to compute an extension C_{new} such that $C \bigcup C_{new}$ is a cutset. **For example**, Fig. 4.2.2(a) illustrates a *BFS* layering of a graph G. Assume that the input configuration $C = \{(f_1, full), (f_2, full), (r_1, reduced)\}$, and $k_{req} = 2$. Here, $L_1 = \{x_1, x_2, f_1\}$ doesn't contain a cutset (since $(f_1, full) \in C$). ■

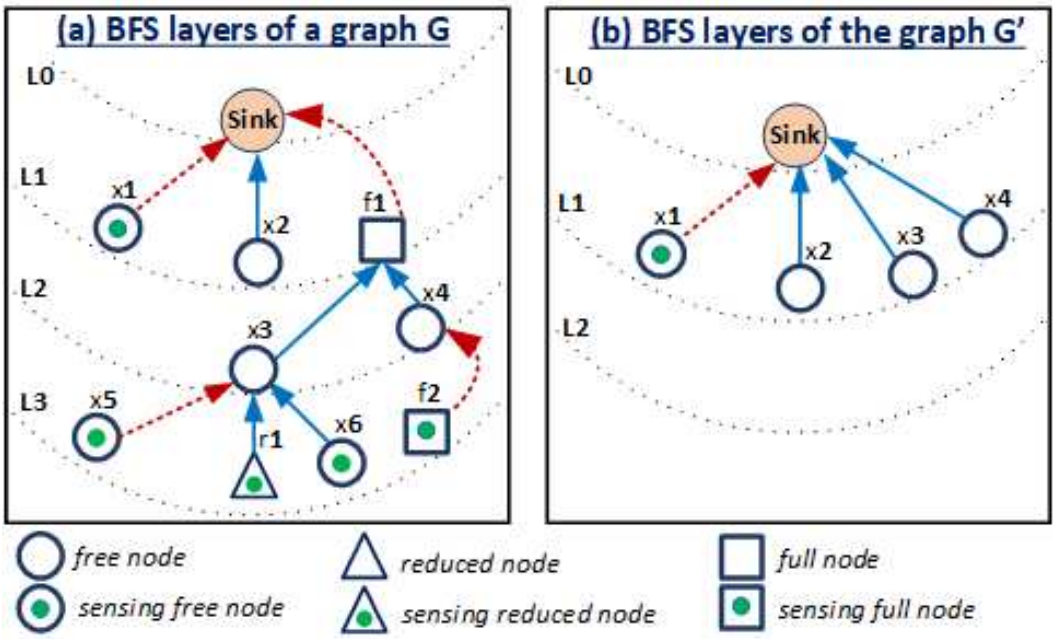


Figure 4.2.2: BFS layers of graphs G and G'

4.2.2.1 Algorithm Idea

Motivated by the possibility of using *BFS* layering to solve the E2C problem, we outline a heuristic algorithm, called E2C-BFS, to solve the E2C problem. The algorithm transforms the input graph G to a directed graph G' where layer L_1 of the *BFS* layering of G' is guaranteed to contain a cutset extension C_{new} if such an extension exists in G. The directed graph G' that satisfies this latter property is constructed iteratively from G. The E2C-BFS algorithm has

two main components:

Firstly, we transform the graph G to G' as follows:

1. Initially, we set $G' = G$

2. Delete from G' all failed nodes in the input configuration C

3. For each node y that is assigned a state $y_s \in \{full, reduced\}$ in C, and y appears on a directed path (x, y, z) where the arc (y, z) exists when y is in state y_s, add an arc (x, z) to G'. Arc (x, z) is a reduced (or full) if arc (x, y) is reduced (respectively, full). After processing all such triplets (x, y, z), delete node y from G'. For example in graph G' of Fig. 4.2.2(a), assume that the input configuration $C = \{(f_1, full), (f_2, full), (r_1, reduced)\}$. After applying step 3 above, node f_1 is deleted from G', and the new arcs $(x_3, sink)$ and $(x_4, sink)$ are added, as shown in Fig. 4.2.2(b).

4. Our devised E2C-BFS algorithm takes the nodes in layer L_1 of G' as failed nodes and uses them as a start point to compute an extension C_{new} (so we set $C_{new} = \{(x, fail) : x \in L_1\}$).

Secondly, the E2C-BFS algorithm uses two additional methods (explained below) to refine the current computed C_{new} to obtain a new configuration with possibly higher occurrence probability $Pr(C_{new})$.

- *Method 1*: This method is motivated by observing that the current computed C_{new} may not be minimal. Thus, *method 1* iterates over every failed node $(x, fail)$ in the current computed C_{new} to see if x can be deleted from C_{new} without affecting the cutset property of the computed solution in the original graph G. If so, $(x, fail)$ is deleted from C_{new}.

- *Method 2*: This method aims at improving the current solution further by iterating over every failed node $(x, fail)$ in the current computed C_{new} where $p_{red}(x) > p_{fail}(x)$ to see if changing $(x, fail)$ to $(x, reduced)$ in C_{new} does not effect the cutset property of the computed solution in

the original graph G. If so, the pair $(x, reduced)$ replaces $(x, fail)$ in the computed solution.

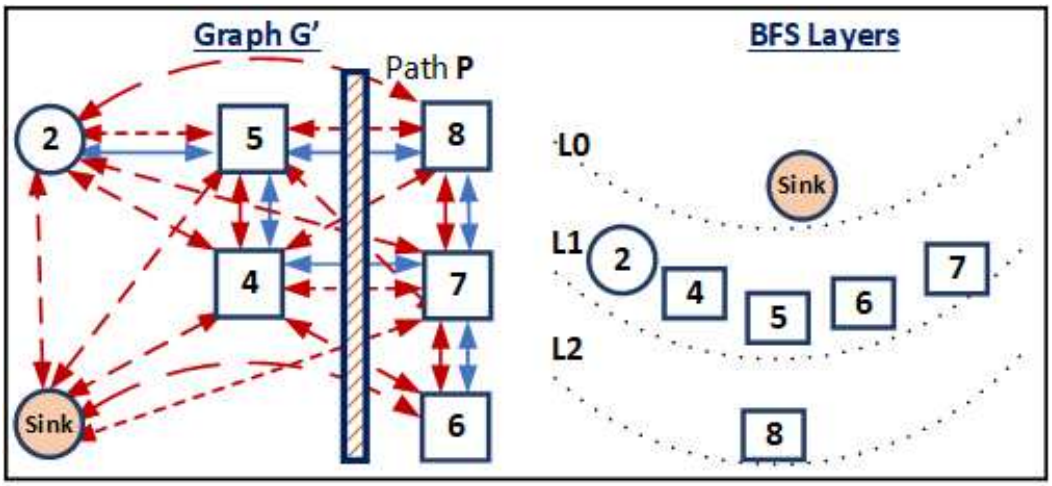


Figure 4.2.3: Construction of G' and its BFS Layers

Example 4.2.2 Fig. 4.2.3 illustrates graph G' constructed from the graph G (shown in Fig. 4.2.1), where the input configuration $C = \{(1, reduced), (3, fail)\}$. Let $k_{req} = 2$, and $C_{new} = \phi$. We do the following:

- Set $G' = G$
- Remove node 3 since $(3, fail) \in C$
- After applying step 3 above, node 1 is deleted from G'; and the new arcs $(2, sink), (4, sink), (5, sink)$ and $(7, sink)$ are added to G' as explained above (note that $(5, 1, sink)$ and $(7, 1, sink)$ are paths in G)
- Obtain a *BFS* layering of G' with arcs directed towards the *sink*. G' is layered into $L_0 = \{sink\}, L_1 = \{2, 4, 5, 6, 7\}, L_2 = \{8\}$
- Set $C_{new} = \{(2, fail), (4, fail), (5, fail), (6, fail), (7, fail)\}$
- Apply *method 1* to each node in C_{new}. Node 2 is removed and we get $C_{new} = \{(4, fail), (5, fail), (6, fail), (7, fail)\}$
- Apply *method 2* to each node in C_{new}. The state of node 6 is changed from $fail$ to $reduced$ (assuming that $p_{red}(6) > p_{fail}(6)$) and $C_{new} = \{(4, fail), (5, fail), (6, reduced), (7, fail)\}$.

■

Function E2C-BFS(G, p, k_{req}, C)
Input: An instance (G, p, k_{req}, C) of the E2C problem where C is an input configuration
Output: If C is extensible, return C_{new}. Else, return -1

1. Set $C_{new} = \phi$
2. Construct G' from G and C as explained above
3. With sink s taken as root, apply *BFS* to divide G' into layers $L_0, L_1, ..., L_K$ where $L_0 = \{s\}$
4. **If** (L_1 is empty): **return -1**
5. **for** ($i = 0$ to $|L_1|$): add $(L_1[i], fail)$ to C_{new}
6. Refine the size of C_{new} using *methods 1*
7. Refine the states of nodes in C_{new} using *methods 2*
8. **return** C_{new}

Figure 4.2.4: Pseudo code for E2C-BFS function

4.2.2.2 Algorithm Details

Function E2C-BFS is summarized in Fig. 4.2.4. Step 1 initializes $C_{new} = \phi$. Step 2 transforms G to a directed graph G' as explained above. Step 3 utilizes *BFS* rooted at the sink s on graph G' to obtain layers $L_0, L_1, ..., L_k$ where $L_0 = \{s\}$. In Step 4, if $L_1 = \phi$, then C is not extensible to a cutset. Else, augment C_{new} with a set of $(node, fail)$ pairs where $|C_{new}| == |L_1|$ as shown in steps 5. Step 6 applies *method 1* on the current computed C_{new} iteratively to minimize the number of candidate pairs in C_{new} without losing the cutset property. Similarly, Step 7 applies *method 2* to change the states of some nodes in C_{new} to the states that have higher probability without losing cutset property. Lastly, step 8 returns the refined extension C_{new}.

4.2.2.3 Correctness and Running Time

Theorem 4.2.1 Let (G, p, k_{req}, C) be an instance of the E2C problem where C is an extensible configuration then:

1. Nodes in L_1 generated by the E2C-BFS algorithm includes an extension C_{new}

2. Applying *methods* 1 and 2 to C_{new} guarantees that C_{new} is minimal and the nodes are assigned better possible states (*reduced* if $p_{red}(x) > p_{fail}(x)$, otherwise, $p_{fail}(x)$).

Proof. Part 1 follows since the sink s in the constructed graph G' is only reachable by nodes in L_1 of the *BFS* tree rooted at s in G'. Thus, having these nodes in C_{new} is sufficient to identify if such extension C_{new} exists.
Part 2 follows since utilizing *method 1* iteratively on candidate node-state pairs in the current computed C_{new} guarantees that any extra pair in C_{new} is removed without violating the cutset property. Similarly, applying *method 2* iteratively on any $(node, state)$ pair in C_{new} guarantees that this node in C_{new} is assigned to a node state with higher probability. ■

Running time: Let $G = (V, E)$ be a directed graph on $n = |V|$ nodes, and $m = |E|$ directed edges. It takes $O(n + m)$ time to do each of the following steps: Step 2 (constructing G'), Step 3 computing L_1, and testing whether a configuration is a cutset (needed in Steps 6 and 7). Since *method 1* (or, 2) tests each node in L_1, it follows that the overall function requires $O(n.(n+m))$ time. ■

4.2.3 The E2C-MaxFlow Algorithm

In this section, we present a better heuristic algorithm to solve the E2C problem. In particular, our devised algorithm, called E2C-MaxFlow, utilizes a solver to the max-flow problem (see, e.g., [17]) to solve the E2C problem. We recall that in the maximum flow problem, we are given a directed graph $G = (V, E)$ with two distinguished vertices: a source s and a destination t, where each arc $(u, v) \in E$ has a non-negative capacity $c(u, v)$. The problem is to find the maximum (s, t)-flow value that satisfies arc capacity and flow conservation constraints. By solving the max-flow problem, we obtain a cut

(i.e., a subset of arcs whose removal disconnect s from t) with minimum total arc capacity.

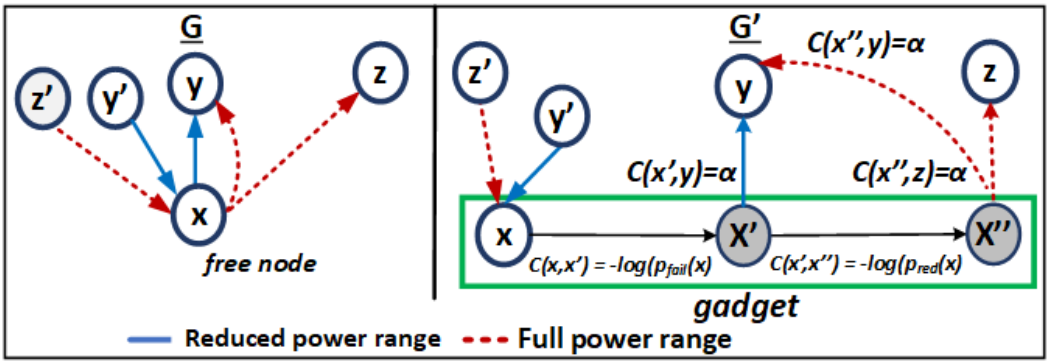


Figure 4.2.5: Replacement of a free node $x \in G$ with a gadget in G'

Our devised algorithm is based on considering a simplified (or restricted) E2C problem. In the simplified problem, we assume that we are given a subset β of sensing nodes (nodes that can sense the intrusion path $\mathbf{P}$) whose disconnection from the sink creates a cutset. We call such a subset β a *cutset generating* (CGEN, for short) set of nodes.

So, if N_{sense} is the number of sensing nodes in G, it suffices to consider CGEN sets of size at most $|\beta| = N_{sense} - k_{req} + 1$ sensing nodes to create an EXPO-RU cutset. We present an algorithm that takes an input configuration C and a subset β of sensing nodes and computes an extension C_{new} such that all nodes in β are disconnected from the sink in $C \bigcup C_{new}$.

The devised E2C-MaxFlow algorithm generates a number of possible CGEN sets, computes an optimal cutset for each β subset, and then chooses the best solution. Thus, the above strategy gives an optimal solution to the E2C problem, if all possible candidates for the set β (at most $\binom{N_{sense}}{N_{sense}-k_{req}+1}$) are considered.

4.2.3.1 Algorithm Idea

Given an instance $(G, C, p, \mathbf{P}, k_{req})$ of the E2C problem, and a CGEN set β ($|\beta| \leq N_{sense} - k_{req} + 1$) of sensing nodes, we present an optimal algorithm to compute C_{new} such that all nodes in β are disconnected from sink in $C \bigcup C_{new}$. In particular, we transform the problem graph G to a directed flow graph G'

such that an optimal solution can be obtained from a minimum capacity arc cut in G'. The flow network G' is constructed as follows. In G', some arcs are assigned a large capacity, denoted α. We set α to be larger than the sum of all arc capacities inside all gadgets of G' (as described below).

1. G' has a new node, denoted s^*, that acts as a source of flows in the max-flow problem. s^* has a directed edge to each node in β.

2. The sink node acts as the destination node t in the max-flow problem.

3. **Free nodes**: in G', we replace each free node x ($x \notin C$) with a gadget, denote $[x, x', x'']$, (shown in Fig. 4.2.5 as green rectangle). In the gadget, we set the following arc capacities: $c(x, x') = -log(p_{fail}(x))$ and $c(x', x'') = -log(p_{red}(x))$. In addition, we have incoming and outgoing arcs of node x.

   - **Incoming arcs to** x: for each incoming arc to x in G, there is an incoming arc to x in the corresponding gadget in G'. The capacity of this arc is set to α.
   - **Outgoing arcs from** x: for each arc (x, y) (or, (x, z)) in G where y (respectively, z) is reachable from x in the reduced (respectively, full) state, G' has an arc (x', y) (respectively, (x'', z)) of capacity α.

4. **Assigned nodes**: for each non-failed node $x \in C$ with incoming (or, outgoing) arc in G, there is a corresponding arc in G' of capacity α.

5. All failed nodes in C are deleted from G'.

We then run the max-flow algorithm between s^* and the sink node to obtain a minimum capacity $(s^*, sink)$-cut.

Example 4.2.3 Consider an instance of the E2C problem composed of the graph G (shown in Fig. 4.2.1), an input configuration $C = \{(1, reduced), (3, fail)\}$ that leaves the remaining 6 non-sink nodes *free*, and $k_{req} = 2$. Since $N_{sense} = 6$ nodes of which node 3 is failed and $k_{req} = 2$, the size of each CGEN set is $5 - k_{req} + 1 = 4$. Thus, we may choose, e.g., a CGEN set $\beta = \{4, 5, 6, 7\}$. The graph G' (shown in Fig. 4.2.6) is obtained from the given instance as follows:

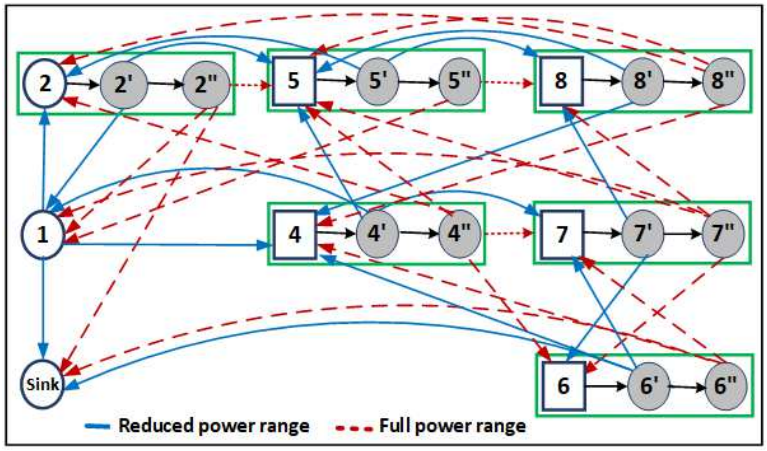


Figure 4.2.6: Construction of G'

- A new node s^* is added to G' (not shown in Fig. 4.2.6 to avoid cluttering the diagram) and it is connected to nodes in β. The node s^* works as a source of flows in a max-flow problem (Step 1 above),
- G' contains the *sink* node that works as the destination node t in the max-flow problem (Step 2 above),
- Each non-sink free node x ($x \notin C$) is replaced with a *gadget* (Step 3 above),
- Since $(1, reduced) \in C$, in G' all incoming (or, outgoing) arcs to node 1 have capacity α (Step 4), and
- Node 3 is removed from G' since 3 has state $fail$ in C (Step 5 above).

■

4.2.3.2 Mapping a Minimum Capacity $(s^*, sink)$-cut to a Solution Configuration C_{new}

We remark that a minimum capacity $(s^*, sink)$-cut either uses exactly one arc from each gadget or no arc from the gadget. If arc (x, x') is used, then we add $(x, fail)$ to the solution C_{new}. Else, if arc (x', x'') is used, then we add $(x, reduced)$ to the solution C_{new}.

Function E2C-MaxFlow(G, p, k_{req}, C)
Input: An instance (G, p, k_{req}, C) of the E2C problem where C is an extensible input configuration
Output: return $C \bigcup C_{new}$

1. Set $C_{new} = \phi$.
2. Construct G' from G and C as explained above
3. Generate the set $\{\beta_0, \beta_1, ..., \beta_r\}$ of all possible CGEN sets
4. Initialize $best\_cut = \phi$.
5. **foreach** β_i in $\{\beta_0, \beta_1, ..., \beta_r\}$
   {
6. Make s^* adjacent to all nodes in β_i
7. Solve an instance of *max-flow* problem on β_i and return $(s^*, sink)$-cut
8. **If** ($(s^*, sink)$-cut of total capacity $< \alpha$ exists):
9. Update $best\_cut$ with the best found $(s^*, sink)$-cut
   }
10. Set $C_{new} = best\_cut$
11. **return** $C \bigcup C_{new}$

Figure 4.2.7: Pseudo code for function E2C-MaxFlow

4.2.3.3 Algorithm Details

Fig. 4.2.7 gives a pseudo code for function E2C-MaxFlow. Step 1 initializes $C_{new} = \phi$. Step 2 obtains G' from G and C as explained above. Step 3 generates a set $\{\beta_1, \beta_2, \beta_3, ..., \beta_r\}$ of all possible CGEN sets. The main loop in step 5 iterates over each set β_i. Step 7 solves an instance of the *max-flow* problem on nodes in β_i where $source = s^*$ and $terminal = sink$. If a mincut exists, update $best\_cut$ as shown in step 9. Step 10 sets $C_{new} = best\_cut$. Step 11 returns $C \bigcup C_{new}$.

4.2.3.4 Correctness and Running Time

In the following argument, we use (G, C) to denote the graph G constrained by the input configuration C, and let G' be the corresponding flow graph constructed by the above procedure. In addition, let $\beta \subseteq N_{sense}$ be any CGEN set considered by the algorithm. For convenience, we use *dipath* to refer to a directed path, and use $(\beta, sink)$-*dipath* to refer to a directed path that links some node in β to the sink. Correctness and optimality aspects of the algorithm

follows from the following properties.

1. For any two non-failed nodes x_1 and x_r, the graph (G, C) has a dipath $P = (x_1, x_2, ..., x_r)$ if and only if the graph G' has a dipath linking x_1 to x_r.

2. The graph (G, C) has a cutset extension C_{new} where the graph $(G, C \bigcup C_{new})$ has no $(\beta, sink)$-*dipath* if and only if G' has a $(s^*, sink)$-cut of capacity $< \alpha$.

3. Let C_{new} be a cutset extension obtained by the above approach so that all nodes in β are disconnected from the sink in $C \bigcup C_{new}$, then $\Pr(C_{new})$ is largest among all such cutset extensions.

Running time: The devised E2C-MaxFlow algorithm gives an optimal solution to the E2C problem if all possible minimal CGEN sets (at most $O(N_{sense}^{k_{req}})$) are processed by the main loop of step 5. The time required by each iteration of step 5 is dominated by the time required to solve a given instance of the maximum-flow problem, denoted $O(T_{MF})$. Reference [17] discusses $O(n^3)$ (and faster) algorithms to solve the maximum-flow problem. Thus, the overall running time of function E2C-MaxFlow is $O(T_{MF}.N_{sense}^{k_{req}})$. ■

4.3 Algorithms for Upper Bounds

Our devised E2C-BFS (c.f. 4.2.2) and E2C-MaxFlow (c.f. 4.2.3) algorithms presented in the previous sections can be used to design a variety of algorithms to compute UBs for the EXPO-RU problem. In this section, we discuss the ideas of designing iteratively and non-iteratively improvable algorithms to compute UBs on exact solutions of the EXPO-RU problem.

4.3.1 An Iteratively Improvable Algorithm

An iteratively improvable UB algorithm produces an exact solution if allowed to run to completion. Else, it outputs an UB on the solution. As discussed in

Chapter 1, an iteratively improvable algorithms typically rely on methods for systematically generating a maximal set of pathsets that suffices to compute the exact solution. The *factoring* method discussed in [7] is an algorithmic framework that serves the above purpose, and the method applies to a broad class of network reliability problems.

The method can be used to compute an UB from a set $\{c_1, c_2, \cdots, c_r\}$ of *s-disjoint* cutsets that can be generated systematically, where an UB can be computed as follows

$$\text{Expo}(G, p, k_{req}) \leq 1 - \sum_{i=1}^{r} Pr(c_i) \tag{4.3.1}$$

One way to ensure that the above equation holds is to ensure that any pair of generated cutsets c_i and c_j have at least one common node, say x, such that c_i and c_j assign two different states to node x. We call such cutsets (or configurations) *s-disjoint*. For the EXPO-RU problem, the two configurations $C_1 = \{(1, reduced), (2, fail)\}$, and $C_2 = \{(1, full), (2, fail)\}$ are s-disjoint since node 1 is assigned two different states in C_1 and C_2.

In the next section, we present UB results obtained using the factoring method. The method converges to computing an exact solution when all possible iterations are done. For the EXPO-RU problem, the factoring method makes at least one call to the E2C-MaxFlow (or, E2C-BFS) algorithm to generate problem cutsets in each iteration. Over all iterations performed, the factoring method outputs an UB that is a sum of disjoint products expression. So, any two configurations C_i and C_j used in the expression are selected such that $\Pr(C_i) + \Pr(C_j)$ is the probability that at least one of C_i and C_j occurs.

4.3.2 A Non-iteratively Improvable Algorithm

As mentioned in Section 1.5, a non-iteratively improvable UB algorithm computes a single value after processing a given problem instance. Effective non-iteratively improvable algorithms are useful both as standalone tools, and also as subroutines called from within an iteratively improvable algorithm to handle some cases.

One such algorithm computes an UB from a given set of pathsets $\{c_1, c_2, \cdots c_r\}$ that are pairwise **node-disjoint**. Computing upper bounds from node-disjoint pathsets has been extensively studied in the context of many network reliability problems where each node can either be *operating* or *failed* (see. e.g., [7]).

In Section 1.2.3, we presented a well-know formula for obtaining an UB from edge-disjoint cutsets for the Rel_2 problems where each element (e.g., edge) can be either operating or failed.

In this section, we use a similar formula to obtain an UB for the 3-state EXPO-RU problem as follows.

1. Compute a set $\{c_1, c_2, ..., c_r\}$ of node-disjoint cutsets
2. For each node x where $(x, reduced)$ appears in a cutset c_i, set the occurrence probability $p_{occ}(x) = p_{red}(x) + p_{fail}(x)$
3. For each node x where $(x, fail)$ appears in a cutset c_i, set the occurrence probability $p_{occ}(x) = p_{fail}(x)$
4. For each cutset $c_i = \{(x, s_x) : \text{node } x \text{ is either reduced or fail}\}$, set $\Pr(c_i) = \prod_{(x,s_x)\in c_i} p_{occ}(x)$
5. Compute an UB on the solution using formula (4.3.2):

$$\Pr[\text{None of the cutsets occur}] = \prod_{i=1}^{r}(1 - Pr(c_i)) \tag{4.3.2}$$

To generate a set of node-disjoint cutsets, we use a simple iterative algorithm in which each iteration calls function E2C-MaxFlow (or, E2C-BFS) to find a cutset (if possible), and then removes its nodes before starting the next iteration. Similar ideas are used in Chapter 3 to compute LBs for the EXPO-RU problem from node disjoint pathsets.

4.4 Numerical Results

In this section, we present numerical results obtained by conducting a set of performance evaluation experiments. The goal is to illustrate the usefulness of

our devised E2C-MaxFlow and E2C-BFS algorithms in obtaining good bounding algorithms as well as exploring the relative strength of the obtained UB algorithms against LB algorithms presented in the Chapter 3. We use the following abbreviations (graph legends) to refer to various LB and UB methods:

- **NDC (or, NDP)**: bounds from node-disjoint cutsets (or, pathsets).
- **Factoring**: UBs (or, LBs) from the factoring algorithm. The UBs are obtained by integrating the factoring algorithm with either the E2C-MaxFlow or E2C-BFS functions discussed in this chapter. The LBs are obtained by integrating the factoring algorithm with the E2P function discussed in Chapter 3. The function E2P is used to extend a given configuration to a pathset (if possible).

To allow for direct comparisons, we use the same network topologies and we run experiments on the class of $W \times W$, where $W \geq 2$, double-diagonal grids (x-grid, for short) where the rows (or, columns) are numbered $0, 1, \cdots W-1$ from left to right (respectively, bottom to top). Fig. 4.2.1 illustrates a 3×3 x-grid. The intrusion path $\mathbf{P}$ is assumed to be placed vertically between the rightmost 2 columns. For simplicity, we assume that all nodes have identical transmission ranges when operating in their reduced (or full) power states, and only nodes placed on the left and right of the path $\mathbf{P}$ can sense the path. The sink node is located at coordinates $(x = 0, y = 0)$. We have implemented all algorithms in Python and run on 2.8 GHz personal computer with 8 GByte memory.

Table 4.1: Exact computations

x-grid size	k_{req}	**Generated configurations (UB)**	**Cutsets found**	**Generated configurations (LB)**	**Pathsets found**
3×3	4	555	309	665	410
3×3	5	983	353	1103	497
4×4	4	17213	7764	23403	12142
4×4	5	32041	11623	46812	14396

4.4.1 Exact Computations

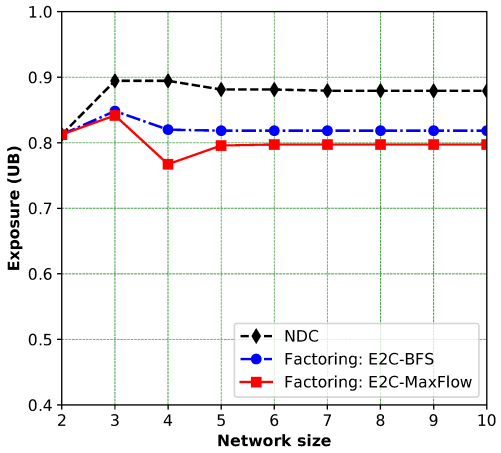
The factoring algorithm mentioned above computes exact results when allowed to run to completion. This is done by generating a maximal set of pairwise s-disjoint configurations (pathsets or cutsets) while avoiding the generation of many useless configurations. Table 4.1 presents the number of generated cutsets (the full implementation uses algorithm E2C-MaxFlow), and the number of generated pathsets (the full implementation uses algorithm E2P) needed to compute $\text{Expo}(G, p, k_{req})$ for 3×3 and 4×4 x-grids. The results show that the number of generated configurations by the factoring algorithm is significantly smaller than the maximum number of possible network states, where the maximum possible number of states for a 3×3 and 4×4 networks are 3^8 and 3^{15}, respectively.

4.4.2 Comparing upper bounds

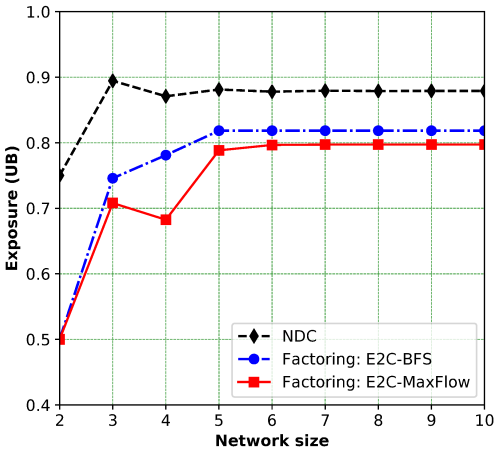
Here, we compare the UBs obtained by NDC, and Factoring (the full implementation uses both the E2C-MaxFlow and E2C-BFS functions). The experiments use x-grid networks of size $W \times W$, where $W \in [2, 10]$, and each non-sink node x has $p_{full}(x) = p_{red}(x) = 0.25$ with $k_{req} = 1$ (Fig. 4.4.1a) and $k_{req} = 2$ (Fig. 4.4.1b). In general, the UBs obtained by the factoring algorithm after performing 1000 iterations is better than the NDP bounds. This is expected as the factoring algorithm generates higher number of cutsets compared to those used in the NDC bounds. Moreover, both Fig. 4.4.1a and Fig. 4.4.1b show that the UBs computed by the E2C-MaxFlow approach achieves strict improvements over bounds from the E2C-BFS as the E2C-MaxFlow approach can give an optimal solution to the E2C problem as explained above.

4.4.3 Gaps between factoring bounds

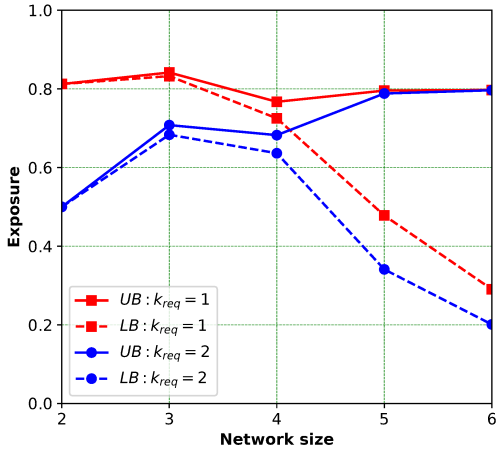
Here, we comment on the gaps between the UBs and the LBs obtained by the factoring algorithm. In particular, Fig. 4.4.1c shows gaps between the UBs obtained by integrating the factoring algorithm with the E2C-MaxFlow function and the LBs obtained by integrating the factoring algorithm with



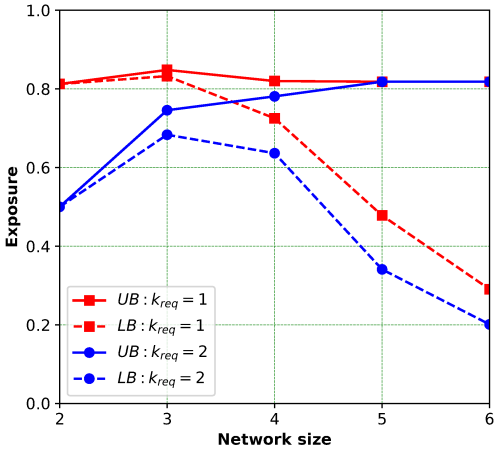
(a) Upper bounds versus network size $k_{req} = 1$



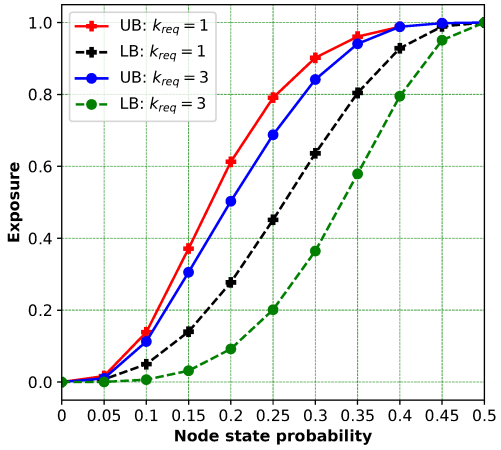
(b) Upper bounds versus network size $k_{req} = 2$



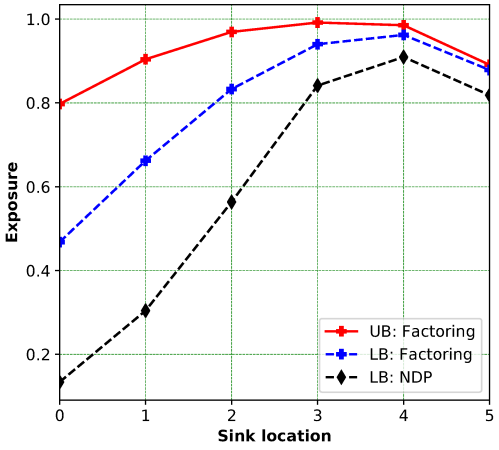
(c) Gaps between obtained bounds using E2C-MaxFlow and E2P



(d) Gaps between obtained bounds using E2C-BFS and E2P



(e) Effect of node state probability



(f) Exposure versus sink location

Figure 4.4.1: The obtained numerical results

the E2P function (presented in Chapter 3). On the other hand, Fig. 4.4.1d shows gaps between the UBs obtained by integrating the factoring algorithm with the E2C-BFS function and the LBs obtained by integrating the factoring algorithm with the E2P function. In the experiments, we use x-grid networks of size $W \times W$, where $W \in [2, 6]$. Also, we set $p_{full}(x) = p_{red}(x) = 0.25$ for all nodes, and the value of $k_{req} \in [1, 2]$. In general, it can be observed that the gaps between the UBs and LBs increase as the network's size increases since only a relatively small number of configurations can be examined when performing 1000 iterations of the factoring algorithm.

We also note that the obtained LBs achieved by the entire network is comparable to the assigned probability $p_{full}(x)$ (or, $p_{red}(x)$) of an individual node. This shows the possibility of building a large network whose performance is at least as good as the performance of its individual nodes.

The obtained results (both UBs and LBs curves) can be used by a network designer to compare the performance of different energy harvesting networks. The low reliability values shown by the lower bound curves indicate to the designer that this type of networks requires careful investigations before deployment.

4.4.4 Exposure versus node state probability

Here we use the UBs and LBs curves obtained by the factoring algorithm (1000 iterations) on 6×6 x-grid network. Fig. 4.4.1e shows the improvement gained in Expo(G, p) when we set $p = p_{red}(x) = p_{full}(x)$ for each node x, and vary p in range $[0.0, 0.5]$ and set $k_{req} = 1, 3$. The results show that the Expo(G, p) increases as p increases. This is expected as we get more reliable configurations that contribute to the obtained bounds.

The results can be used by a network designer to explore the quality of Expo(G, p) by finding a minimum value, denoted p_{min}, for which Expo$(G, p) \geq p_{min}$. Using the shown LBs curves, one can estimate $p_{min} \approx 0.2$ and $p_{min} \approx 0.3$ for $k_{req} = 1$ and $k_{req} = 3$, respectively. The shown results are encouraging since a network designer expects the performance of the overall network to exceed the performance of each single node over a wide range of node operating

parameters.

Moreover, in the context of deployment strategies, if the size of the surveillance area increases, the designer might need to increase k_{req} to avoid false alarms and increase detection probability. The curves illustrate that the Expo(G, p) decreases as k_{req} increases (i.e., $k_{req} = 3$). This is expected since configurations with a higher number of sensing nodes, required to monitor an intrusion path $\mathbf{P}$, have more nodes, and hence contribute less to the obtained bounds

4.4.5 Identify an optimal location of the sink node

An interesting problem for an EH-WSN network designer is to identify a best location of the sink node that maximizes the overall network reliability. Here, we use the LB and UB curves to identify a best location of the sink that maximizes the exposure measure. We use a 6×6 x-grid network where the intruder's path $\mathbf{P}$ passing vertically between the last two columns, and the sink node is located on the diagonal of the network at locations $(0, 0), (1, 1), ..., (5, 5)$. We set $p_{full}(x) = p_{red}(x) = 0.25$ for all nodes and set $k_{req} = 1$. Fig. 4.4.1f illustrates the obtained bounds.

Despite the differences between the obtained bounds, the curves show that the optimal location of the sink node is at $(4, 4)$-coordinate. This location wins for its closeness to the intruder path $\mathbf{P}$, and hence the quality of the obtained cutsets and pathsets at this location is better than in other locations.

4.5 Concluding Remarks

In this chapter, We formulate the E2C problem for the EXPO-RU problem. Then we propose two efficient heuristic algorithms to solve the E2C problem. The first devised algorithm, called E2C-BFS, utilizes the breadth-first-search technique while the second one, called E2C-MaxFlow, utilizes a solver to the max-flow problem. The key strength of the second algorithm compared to the first one is that it can produce an optimal solution for E2C problem if it runs for termination.

Then, both E2C-BFS and E2C-MaxFlow algorithms are used to design iteratively and non-iteratively improvable UB algorithms for the EXPO-RU problem. Numerical results obtained to explore the relative strength of the devised UB algorithms against LB algorithms presented in Chapter 3.

Chapter 5

Bounding EXPO-RU Using Dynamic Programming Approach

In this chapter, we consider two general types of reliability problems on probabilistic graphs where each node can be independently in one of a possible number of node states (such as the EXPO-RU problem). Each type has two problems defined on pathsets and cutsets, respectively.

In the cutsets version of the first problem type, we are given a sequence Q of network cutsets, and we want to compute an UB on the exact solution by computing the probability that no cutset in Q occurs. The pathset version is defined similarly to compute a LB. We develop a dynamic programming approach to compute such probability, assuming that node states satisfy a coherence property. In general, the running time of the devised approach grows exponentially with the size of the input set Q. We show, however, that when Q is a disjoint set of cutsets, or a consecutive set, the running time is comparable to the running time of an algorithm that is aware that the input has this property.

In the cutset version of the second problem type, we are given a disjoint set Q of cutsets, and we seek to extend it to a set of consecutive cutsets that exploits the multi-state node model.

Some of the results in this chapter appear in [9, 10].

5.1 System Model

In this section, we briefly review the used network model for the for the EXPO-RU problem as well as important definitions and assumptions.

5.1.1 Network Model

We adopt the same network model used in Chapters 3 and 4. The main aspect of the model is reviewed below. We consider EH-WSNs that rely on energy harvesting (i.e., solar energy). It is assumed that the system time is slotted into a set of time slots and the harvesting rate of each non-sink node might be different by the end of each time slot, and the sink node is assumed to have unlimited energy. Based on stored energy level of each node, fluctuations in a node's energy level affect its communication, sensing and processing. However, since wireless transmission is the most power consuming activity in many EH-WSNs, we assume that a reduction in a node's available energy affects primarily its transmission range. It is also assumed that there is an energy management unit (EMU) that controls states of each node based on its available stored energy level, and it controls the node's transmission ranges. So, we model the fluctuations in a node's stored energy by associating a probability that a node works either in full power, reduced power, or node can not work at all when its energy is depleted.

In particular, we propose a model that uses in its basic form the following 3-state node model for each non-sink node x in the network:

- If x's stored energy level lies within a designer specified range (e.g., say $[70\% - 100\%]$), x is assumed to be able to communicate in full transmission power; then x is considered to be in the $full$ energy state (with probability $p_{full}(x)$).

- Else, if x's stored energy level lies within a lower range (e.g., say $[30\% - 70\%]$), x is assumed to be communicating with reduced transmission power; then x is considered to be in the *reduced* energy state (with probability $p_{red}(x)$).

- Else, x has no enough energy to communicate, then x is in the $fail$ state with failure probability $p_{fail}(x) = 1 - (p_{full}(x) + p_{red}(x))$.

The overall EH-WSN is modelled by a probabilistic directed graph $G = (V \bigcup \{s\}, E)$ where V is a set of nodes in G, s is a distinguished sink node, and E is a set of directed edges. It is assumed that the sink node s does not perform sensing tasks and it is not subject to power fluctuations. For any non-sink node $x \in G$, x can be in a state $s_x \in \{full, reduced, fail\}$ during a period of time when the network G is analyzed.

5.1.2 Review of the EXPO-RU Problem

The EXPO-RU problem (defined in Section 3.1) uses a probabilistic graph (G, p) to model the network, and the problem formulation assumes that different nodes behave independent of each other. During a short random time interval, the network G is in some **network state** S where each node x is in some state $s_x \in \{full, reduced, fail\}$. We use $S = \{(x, s_x) : x \in V, s_x \in \{full, reduced, fail\}\}$ to refer to any such network state. The probability that a given network state S arises is $\Pr(S) = \prod_{(x,s_x)\in S} p(x, s_x)$. Each network state S is either *operating* or *failed*. To be operating, node states in S should be such that at least k_{req} sensing nodes in S can reach the sink node. Else, S is in a *failed* state. The EXPO-RU problem is called to compute the probability that the network G is in an operating state S. We denote such probability by $\text{Expo}(G, p, \mathbf{P}, k_{req})$, or $\text{Expo}(G, p)$ for short.

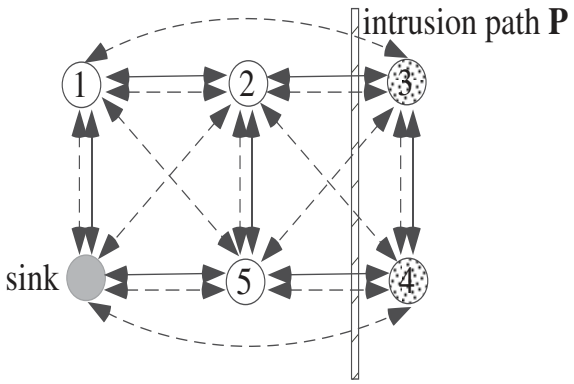


Figure 5.1.1: An instance of the EXPO-RU problem

Example 5.1.1 Fig. 5.1.1 illustrates an instance of the EXPO-RU problem. Dashed (respectively, solid) links represent full (respectively, reduced) transmission ranges. Each double-arrowed link represents two independent arcs, one in each direction (for simplicity, we assume that if arc (x, y) exists then arc (y, x) exists). Dotted circles represent nodes that can sense **P**. ■

5.1.3 Concepts Needed for Algorithms

In addition to the concepts of 3-state node model, network state, network configuration, pathsets, and cutsets introduced in previous chapters; we need the following definition.

Coherence: A reliability system is *coherent* if every superset of a pathset (or a cutset) is a pathset (respectively, cutset) [16]. The majority of results in the literature apply to coherent systems (such as the EXPO-RU problem).

The stronger (weaker) than relation: In addition, we also need the following relation on the set of possible node states $\{s_1, s_2, \cdots, s_{last}\}$: for two different states s_{st} and s_{wk}, s_{st} is *stronger* than $(>)$ s_{wk} (conversely, s_{wk} is *weaker* than $(<)$ s_{st}) if replacing the node-state pair (x, s_{wk}) with (x, s_{st}) in any pathset gives a pathset, and replacing (x, s_{st}) with (x, s_{wk}) in any cutset gives a cutset. Our algorithms deal with problems where the set of node states under the strength relation is a **total order**. In the EXPO-RU problem a state with a longer transmission range is stronger than a state with a shorter transmission range.

5.2 Bounds from Cutsets and Pathsets

The main focus of our work here is on developing algorithms for computing upper bounds (UBs) and lower bounds (LBs) from given sequences of cutsets and pathsets, respectively. We aim at developing algorithms that:

(a) are conceptually simple,

(b) handle problems with multistate nodes,

(c) work with any arbitrary sequence Q of cutsets or pathsets (no particular structure is required),

(d) allow each node to appear in any state in each cutset or pathset,

(e) runs in polynomial time in the number of nodes used in Q, and the number of possible node states, and

(f) exhibit substantial improved execution times when processing structured sequences for which efficient algorithms exist.

For a brief related background, we note that the direction of obtaining bounds from cutsets and pathsets is a fundamental direction that has received extensive research work (see, e.g., [7, 14, 16]). When $Q = (C_1, C_2, \cdots, C_r)$ is a given sequence of cutsets (respectively, pathsets), the desired UB is given by Pr(*no cutset in* Q *occurs*) (respectively, the desired LB is given by Pr(*at least one pathset in* Q *occurs*)). If Q contains all possible mincuts (respectively, minpaths) of a problem instance, the UB (respectively, LB) is the exact solution to the instance. Since some #P-complete reliability problems on special classes of networks have polynomial number of mincuts (or, minpaths), it is unlikely that a general algorithm exists that can handle any set Q of cutsets (or, pathsets) in time polynomial in the size of Q.

Efficient algorithms for computing the desired bounds exist when the sequence Q has some special structure. For example, if all cutsets (or pathsets) in Q are pairwise disjoint then a simple formula can be used to compute the desired bounds (see, e.g., [8] for the EXPO-RU problem).

Additionally, if the sequence Q satisfies a property called the *consecutive* sets property then also efficient solutions exist (see, e.g., [49]). The consecutive property holds if whenever a node x belongs to two sets C_i and C_j, $i < j$, then it belongs to all intermediate sets C_k, $i \leq k \leq j$.

Our developed algorithms here satisfy the above desired properties, and work for any coherent system where the node states have a total order under the strength relation.

5.3 A Cutsets Algorithm for Multistate Nodes

In this section, we present our general algorithm for computing the probability that no cutset in a given sequence $Q = (C_1, C_2, \cdots, C_r)$ of cutsets occurs. That is, the algorithm takes as an input a cutset sequence $Q = (C_1, C_2, \cdots, C_r)$, and a sequence $X = (x_1, x_2, \cdots, x_{n_Q})$ and produces the output Pr(*no cutset in* Q *occurs*). To achieve efficiency, we use a dynamic programming approach that stores and updates information about many network configurations in one dynamic program configuration type, as described below. We first introduce the two following ingredients:

- Node processing order, and
- Dynamic program configurations types.

5.3.1 Node Processing Order

Any ordering $X = (x_1, x_2, \cdots, x_{n_Q})$ of the nodes used in the sequence $Q = (C_1, C_2, \cdots, C_r)$ gives a correct solution. However, improved bounds on the worst case running times can be shown in some special cases if we use the following ordering, called *left-right* (LR) ordering. For a node x, denote by $first(x)$ ($last(x)$) the index $i \in [1, r]$ of a cutset C_i where node x is first (respectively, last) used in the input sequence Q. A sequence X satisfies the LR ordering property if node x_i precedes node x_j, $i \leq j$, in X only when either **(a)** $first(x_i) \leq first(x_j)$, or **(b)** $first(x_i) = first(x_j)$ and $last(x_i) \leq last(x_j)$.

	C_1	C_2	C_3	C_4
w	*reduced*	*fail*		
x		*reduced*	*fail*	
y		*reduced*		*fail*
z			*reduced*	

Figure 5.3.1: Incidence relation between 4 cuts and 4 nodes

Example 5.3.1 Fig. 5.3.1 illustrates the incidence relation between a sequence $Q = (C_1, C_2, C_3, C_4)$ of 4 cutsets and 4 nodes where, e.g., $C_1 = \{(w, reduced)\}$, and $C_2 = \{(w, fail), (x, reduced), (y, reduced)\}$. For node

y, we have $first(y) = 2$ since C_2 is the first cutset in Q that uses y, and $last(y) = 4$. The node ordering $X = (w, x, y, z)$ satisfies the LR property. ■

5.3.2 Dynamic Program Configurations Types

Given a node processing order $X = (x_1, x_2, \cdots, x_{n_Q})$, the algorithm processes node x_i in the ith iteration by examining the new configurations generated by adding x_i in each of its possible states (e.g., the $full$, $reduced$, and $fail$ states in the EXPO-RU problem) to network configurations over the previously processed nodes $(x_1, x_2, \cdots, x_{i-1})$.

The algorithm achieves its efficiency by aggregating information about several network configurations into one equivalence class. Each class corresponds to a particular *configuration type*. The definition uses the assumed total ordering of the possible states from the *strongest* to the *weakest*: (e.g., $(full, reduced, fail)$).

> **Definition (configuration types (c-types for short)).** Let S be a network configuration over nodes $(x_1, x_2, x_3, \cdots)$ processed thus far. The type of S, denoted $type(S)$, is a binary sequence $(b_1, b_2, \cdots, b_r)$ where $b_i = 1$ if at least one node x that appears in cutset C_i appears in S in a state *stronger* than its state in C_i. Else, we set $b_i = 0$ (an initial value). Thus, we set $b_i = 1$ if and only if cutset C_i does not occur in network configuration S. ■

Below, we use $s(x, S)$ (and, (s, C_i)) to denote the state of node x in the configuration S (respectively, cutset C_i).

Example 5.3.2 For the example in Fig. 5.3.1, suppose that the algorithm processes the nodes in the order $X = (w, x, y, z)$. Assume that the algorithm has finished processing nodes w and x. So, it has examined configurations over these two nodes only. For a configuration $S = \{(w, full), (x, reduced)\}$, we have $type(S) = (b_1 = 1, b_2 = 1, b_3 = 1, b_4 = 0)$. Here,

- $b_1 = 1$ since $(s(w, S) = full) > (s(w, C_1) = reduced)$,

- $b_2 = 1$ since $(s(w, S) = full) > (s(w, C_2) = fail)$,
- $b_3 = 1$ since $(s(x, S) = reduced) > (s(x, C_3) = fail)$.
- However, $b_4 = 0$ since there is no node v in C_4 where $s(v, S) > s(v, C_4)$.

■

5.3.3 Overview of the Algorithm

The overall algorithm takes as input a cutset sequence $Q = (C_1, C_2, \cdots, C_r)$, and a sequence $X = (x_1, x_2, \cdots, x_{n_Q})$ and produces the output Pr(*no cutset in* Q *occurs*). The algorithm utilizes two associative array, denoted R and T. Each array provides a key-value mapping, where each key $B = (b_1, b_2, \cdots, b_r)$ is a c-type, and the corresponding value $R[B]$ (or $T[B]$) evolves iteratively to the probability of obtaining network configurations of type B. The algorithm is organized around the following functions.

Function Main. The pseudo code of function main is shown in Fig. 5.3.2. Step 1 initializes the table R to hold one state type $B = (b_1 = 0, b_2 = 0, \cdots, b_r = 0)$ that corresponds to the empty network configuration where no node is assigned to any state. Step 2 iterates over each node x in the input sequence X. In the ith iteration, $i \in [1, n_Q]$, we use node $x = x_i$ to generate all possible c-types where each c-type B' is associated with network configurations over nodes in $(x_1, x_2, \cdots, x_i)$. During each such iteration, table R is used as a source, and table T is used as a destination.

To generate such c-types, the two nested loops in Steps 3 and 4 iterate over all keys in table R, and all possible states $s(x)$ of node x. Step 5 computes the new c-type B' by invoking function NextState, and updating the corresponding probability $T[B']$. Step 6 deletes from table T c-types that do not contribute to the final solution (as explained in Section 5.3.4). Step 7 exchanges the roles of table R and T prior to the start of the next iteration that processes node x_{i+1}. Finally Step 8 returns the computed solution.

Function NextState. The pseudo code of function NextState is shown in

Function Main(Q, p, X)**:**
Input: A sequence $Q = (C_1, C_2, \cdots, C_r)$, $r \geq 1$, of mincuts, a table p of node state probabilities, and a sequence $X = (x_1, x_2, \cdots, x_{n_Q})$ of node processing order for the nodes used in Q

Output: Return $sol =$ Pr(no mincut in Q occurs)

1. initialize $R[(b_1 = 0, b_2 = 0, \cdots, b_r = 0)] = 1.0$
2. **foreach** (node x in the ordering X)
   {
3. **foreach** (state type $B = (b_1, b_2, \cdots, b_r)$ in R)
   {
4. **foreach** (node state $s(x) \in \{full, reduced, fail\}$)
   {
5. a. $B' =$ NextState$(B, x, s(x))$

   b. $T[B'] \mathrel{+}= R[B] \times p_{s(x)}(x)$
   }
   }
6. delete from table T any non-extensible c-type B'
7. exchange pointers to tables R and T; empty table T
   }
8. **return** $sol = R[(b_1 = 1, b_2 = 1, \cdots, b_r = 1)]$

Figure 5.3.2: Function Main for the EXPO-RU problem

Function NextState$(B, x, s(x))$**:**
Input: A state type $B = (b_1, b_2, \cdots, b_r)$, a node x, and a state $s(x)$ of node x

Output: A state type $B' = (b'_1, b'_2, \cdots, b'_r)$ obtained by adding the node-state pair $(x, s(x))$ to any network configuration (on the nodes of X processed thus far) of type B

Notation: $s(x) \leq s(x, C_j)$: input state $s(x)$ equals to, or weaker than, the state of node x in cutset C_j

1. **for** $(j = 1, 2, \cdots, r)$
   {
2. **if** $(x \in C_j$ AND $s(x) > s(x, C_j))$ $b'_j = 1$
   else (i.e., $x \notin C_j$ OR $(x \in C_j$ AND $s(x) \leq s(x, C_j))$ $b'_j = b_j$

   }
3. **return** state type $B' = (b'_1, b'_2, \cdots, b'_r)$

Figure 5.3.3: Function NextState

Fig. 5.3.3. The function takes as input a c-type $B = (b_1, b_2, \cdots, b_r)$ and a node x in state $s(x)$. The function returns a new c-type $B' = (b'_1, b'_2, \cdots, b'_r)$ such

that if S is a configuration of type B then $type(S \bigcup (x, s(x)) = B'$.

This is done by considering the effect of $(x, s(x))$ on each bit b_j, $j \in [1, r]$, in B. In particular, if $b_j = 0$, and $s(x)$ is a stronger state than the state $s(x, C_j)$ of x in C_j then we set $b'_j = 1$. Else, we set $b'_j = b_j$.

5.3.4 Deleting Non-extensible c-types

To further increase the efficiency of the algorithm, Step 6 in function Main deletes the c-types from array T that do not contribute to the final solution associated with the c-type $(b_1 = 1, b_2 = 1, \cdots, b_r = 1)$. We call such c-types *non-extensible*. To explain this step, we introduce the following partitioning of the cutsets in Q into three disjoint subsets during the processing of any node x_i, $i \in [1, n_Q]$ (each subset can possibly be empty):

- $Pre(x_i, X, Q)$: the subset of Q where each cutset C_j has all of its nodes $V(C_j)$ occurring before node x_i in the processing sequence X.
- $Active(x_i, X, Q)$: the subset of Q where each cutset C_j either has node $x_i \in C_j$, and/or C_j has at least one node occurring before x_i in X, and another node occurring after x_i in X.
- $Post(x_i, X, Q)$: the subset of Q where each cutset C_j has all of its nodes $V(C_j)$ occurring after x_i in X.

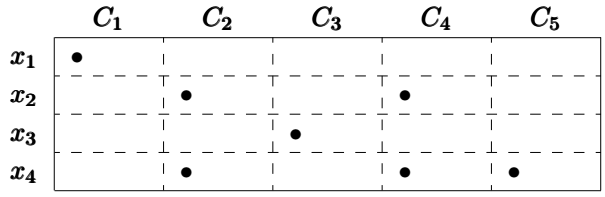


Figure 5.3.4: An example with $r = 5$ cutsets on $n_Q = 4$ nodes

Example 5.3.3 Fig. 5.3.4 illustrates the incidence relation between a sequence $Q = \{C_1, C_2, \cdots, C_5\}$ of cutsets and 4 nodes. For the sequence $X = (x_1, x_2, x_3, x_4)$, we have: $Pre(x_3, X, Q) = \{C_1\}$, $Active(x_3, X, Q) = \{C_2, C_3, C_4\}$, and $Post(x_3, X, Q) = \{C_5\}$.■

As can be seen, during the ith iteration, if $B = (b_1, b_2, \cdots, b_r)$ is a c-type in table R, calling function NextState can return a new c-type B' where $b'_j \neq b_j$ only if the corresponding cutset $C_j \in Active(x_i, X, Q)$. Otherwise (if $C_j \in Pre(x_i, X, Q)$ or $C_j \in Post(x_i, X, Q)$), the function keeps $b'_j = b_j$. Since for any index i, $i \in [1, n_Q]$, we have $Pre(x_i, X, Q) \subseteq Pre(x_{i+1}, X, Q)$, we can characterize non-extensible c-types as follows.

> **Definition (non-extensible c-types).** Given sequences Q and X, during the ith iteration, in Step 6 a c-type $B' = (b'_1, b'_2, \cdots, b'_r)$ in table T is non-extensible (w.r.t. the remaining nodes $(x_{i+1}, x_{i+2}, \cdots)$) if there is an index j, $j \in [1, i]$, such that $b_j = 0$ and the corresponding cutset $C_j \in Pre(x_i, X, Q)$. ■

By the above observation, all such non-extensible c-types can be deleted in Step 6.

Example 5.3.4 Using the example of Fig. 5.3.4, in the iteration that processes node x_3, Step 6 can delete the c-type $B = (b_1 = 0, b_2 = 1, b_3 = 1, b_4 = 1, b_5 = 0)$ since $C_1 \in Pre(x_3, X, Q)$. ■

5.3.5 Correctness and Running Time

Using induction on the number of iterations done by the main loop of function Main, one can show that

Theorem 5.3.1 Following the end of the ith iteration, $i \in [1, n_Q]$, of the main loop in function Main (Step 2) the function computes: **(a)** a complete set of c-types corresponding to network configurations on the prefix $(x_1, x_2, \cdots, x_i)$ of X that contribute to the final result, and **(b)** for each computed c-type B, $R[B]$ is the probability of obtaining a configuration S (over $(x_1, x_2, \cdots x_i)$) where $type(S) = B$. ■

Proof.

Part **(a)** follows since we induct on the iteration number i. The basis for $i = 1$ holds since Step 1 initializes table R with the c-type $(b_1 = 0, b_2 =$

$0, \cdots, b_r = 0)$ corresponding to the empty network configuration over the nodes in $(x_1, x_2, \cdots, x_{n_Q})$. Iteration $i = 1$ generates all possible c-types resulting from taking node x_1 in each of its possible states.

For $i > 1$, we assume that the algorithm satisfies the induction hypothesis for all iteration in the range $[1, i-1]$. The ith iterations starts with a complete and correct table R. The body of the loop exhaustively generates in table T all possible c-types that can be obtained from all c-types in table R by considering node x_i in all of its possible states. Subsequently, Step 6 deletes non-extensible states from table T.

Part **(b)** follows since the algorithm considers the effect of each node state $s(x)$ of each node x in the ordering X on the state types in R.

■

Running Time. The maximum size of table R (or T) determines the worst case running time of the overall algorithm. When $|Q| = r$ cutsets, a rough bound on such a maximum size is 2^r keys (c-types). However, a tighter bound on the size of table R at the end of the ith iteration can be obtained by considering the number of cutsets in the set $Active(x_i, X, Q)$. So, we introduce the following function:

- $n(X, Q)$ is the cardinality of the largest active set among the sets in $\{Active(x_i, X, Q) : i \in [1, r]\}$. For instance, in the example of Fig. 5.3.4, $n(X, Q) = 3$ since $Active(x_3, X, Q) = \{C_2, C_3, C_4\}$ is the largest active set.

Thus, in any iteration, the maximum size of table R is bounded by $2^{n(X,Q)}$.

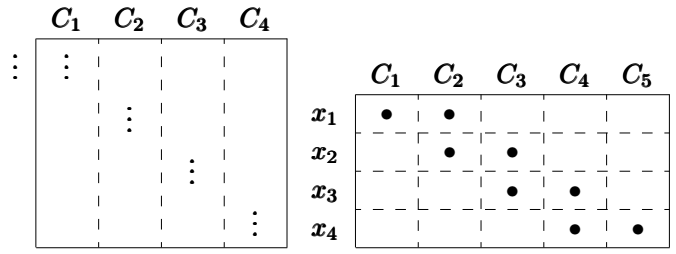
We now derive the running time of the overall algorithm using the following notation: n_Q $(= |X|)$, r $(= |Q|)$, $2^{n(X,Q)}$ (maximum size of table R), $O(n_Q \cdot r)$ (time to compute $n(X, Q)$), and $O(r)$ (time to execute function NextState). We assume a constant number of node states, and the use of associative arrays that provide $O(1)$ access and update time for each entry.

Theorem 5.3.2 Function Main runs in $O(n_Q \cdot 2^{n(X,Q)} \cdot r)$ time.

Proof. The running time is dominated by the execution of Step 5 over all $O(n_Q \cdot 2^{n(X,Q)})$ iterations done by Steps 2 to 4. The result follows since Step 5 requires $O(r)$ time. ■

5.4 Performance on Special Cases

In this section, we show substantially improved running times of the algorithm when processing two special types of cutset sequences. In the first type, Q, $|Q| = r$, is a sequence of pairwise node-disjoint cutsets. For this type, a formula for computing the solution in $O(n_Q)$ time exists. For such a sequence Q, when X satisfies the LR property (cf., Sec. 5.3.1), the layout of the (Q, X)-incidence matrix is as illustrated in Fig. 5.4.1(a). Here, for each node x_i, we have $|Active(x_i, X, Q)| = 1$. Hence, $n(X, Q) = 1$, and the running time reduces to $O(n_Q \cdot r)$.



(a) (b)

Figure 5.4.1: (Q, X)-incidence relations

In the second type, Q is a consecutive sequence of cutsets, and each node x appears in Q in only one state, denoted $s(x, Q)$. For this type, algorithms exist in the literature for computing the solution in $O(n_Q \cdot r)$ time for 2-state (operate/fail) systems. For such a sequence Q, when X satisfies the LR property, the layout of the (Q, X)-incidence matrix is as illustrated in Fig. 5.4.1(b) (each node x appears in a contiguous interval of cutsets in Q).

Here, we note that at the start of the ith iteration, $i \geq 2$ of the algorithm's main loop, table R contains only c-types that matches the pattern 1*0* composed of a (possibly empty) sequence of 1s, followed by a (possibly empty) sequence of 0s. Only c-types matching this pattern can be extended to the

c-type solution ($b_1 = 1, b_2 = 1, \cdots, b_r = 1$) by processing the remaining nodes $(x_i, x_{i+1}, x_{i+2}, \cdots)$, given that any such node x_j appears only in one state $s(x_j, Q)$. The pattern gives rise to at most r c-types. Thus, the maximum size of the table R is r, and the overall algorithm runs in time $O(n_Q \cdot r^2)$.

5.5 A Pathsets Algorithm for Multistate Nodes

The algorithm for computing a LB from a given sequence $Q = (P_1, P_2, \cdots, P_r)$ of pathsets is similar to the cutsets algorithm with the following few (but critical) modifications:

- Function Main, Step 1: initialize $R[(b_1 = 1, b_2 = 1, \cdots, b_r = 1)]$ (this c-type corresponds to assuming that each pathset occurs)
- Function Main, Step 8: **return** $sol = 1 - R[(b_1 = 0, b_2 = 0, \cdots, b_r = 0)]$ (the c-type $(b_1 = 0, b_2 = 0, \cdots, b_r = 0)$ corresponds to configurations where all pathsets do not occur)
- Function NextState, Step 2:
  if ($x \in C_j$ AND $(s(x) < s(x, C_j))$ $b'_j = 0$ **else** $b'_j = b_j$

5.6 Consecutive sets

Algorithms for computing bounds (both UBs and LBs) from a given sequence of cutsets and pathsets are valuable tools for evaluating the reliability of many networks problems. One important class of such algorithms concerns the use of sequences that satisfy the consecutive sets property [44]. Formally, given a set of elements (e.g., nodes or edges of a graph) $E = \{e_1, e_2, ..., e_n\}$ and a sequence $Q = (q_1, q_2, ..., q_r)$ where each q_i is a subset of E, the sequence Q is said to have the consecutive set property **if and only if** whenever an element x belongs to sets q_i and q_k, $i < k$, then x belongs to each intermediate set q_j where $i \leq j \leq k$. In the literature, it has been noted that any sequence of node disjoint sets satisfies the consecutive sets property. Hence, bounds from consecutive sets can improve on bounds obtained from disjoint sets (e.g., [16]).

Examples of using consecutive sets include the work of [45] for computing LBs from pathsets, and [19] for computing UBs from cutsets.

In this section, we present a core algorithm that can be used to build such a consecutive sequence of cutsets or pathsets. The algorithm is an efficient heuristic algorithm to solve a problem, called the consecutive sequence extension (CS_extend) problem, defined below.

> **Definition (the CS_extend problem)**: Given a set of elements $E = \{e_1, e_2, ..., e_n\}$ and a consecutive sequence (of type pathsets or cutsets) $Q = (q_1, ..., q_i, q_{i+1}, ..., q_r)$ of length $|Q| = r$, extend (if possible) Q to a larger consecutive sequence (of the same type as Q of either pathsets or cutsets) $Q' = (q_1, ..., q_i, X, q_{i+1}, ..., q_r)$ of length $|Q'| = r + 1$, where $X \subseteq E$ is a new added set. ■

To simplify the presentation, we henceforth assume that Q is a sequence of cutsets (the solution approach is similar for pathsets).

5.6.1 Algorithm Idea

Our approach to solve the above problem is organized around two functions:

1. CS_extend: (see, Fig. 5.6.3) a function that attempts to compute a new set X that can be inserted in the sequence $Q = (q_1, ..., q_i, q_{i+1}, ..., q_r)$ between q_i and q_{i+1}, for a given input index i, where $i \in [1, |Q| - 1]$, and

2. Q_extend: (see, Fig. 5.6.4) a function that calls the above CS_extend function while varying the index i, $i = 1, 2, ..., |Q| - 1$.

Notation: We need the following notation to explain the function CS_extend. Given a sequence of consecutive cutsets $Q = (c_1, ..., c_i, c_{i+1}, ..., c_r)$, Fig. 5.6.1 sketches the relationship between some of the used notation.

- $E(Q_{sub})$: If Q_{sub} is a (possibly empty) subsequence of Q then $E(Q_{sub})$ denotes the set of elements used in Q_{sub}

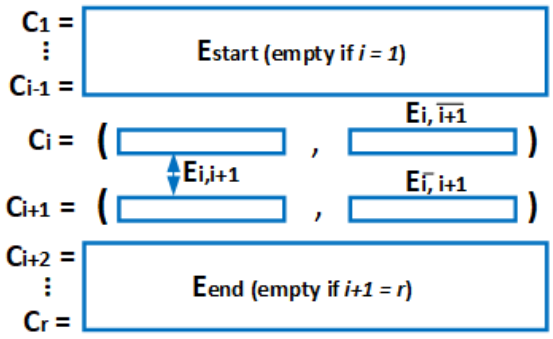


Figure 5.6.1: Notation used in function CS_extend

- E_{start}: The set of elements (possibly empty) used in the prefix $(c_1, ..., c_{i-1})$ that are not used anywhere else in Q. That is, $E_{start} = E((c_1, ..., c_{i-1})) \setminus E((c_i, ..., c_r))$

- $E_{i,i+1}$: The set of elements (possibly empty) used in both c_i and c_{i+1}. That is, $E_{i,i+1} = E((c_i, c_{i+1}))$

- $E_{i,\overline{i+1}}$: The set of elements (possibly empty) used in c_i but not in c_{i+1}. That is, $E_{i,\overline{i+1}} = E(c_i) \setminus E(c_{i+1})$

- $E_{\overline{i},i+1}$: The set of elements (possibly empty) used in c_{i+1} but not c_i. That is, $E_{\overline{i},i+1} = E(c_{i+1}) \setminus E(c_i)$

- E_{end}: The set of elements (possibly empty) used in the suffix $(c_{i+2}, ..., c_r)$ that are not used anywhere else in Q. That is, $E_{end} = E((c_{i+2}, ..., c_r)) \setminus E((c_1, ..., c_{i+1}))$

- E_{rem}: The set of elements (possibly empty) not in Q. That is, $E_{rem} = E \setminus E(Q)$.

To illustrate the use of the above notation, we draw an example using the well-known (s, t)-terminal reliability problem [16]. The problem assumes that edges of a given undirected graph can fail independently of each other. A pathset is a set of edges connecting two distinguished nodes (s and t). A cutset is a set of edges whose removal disconnects s from t.

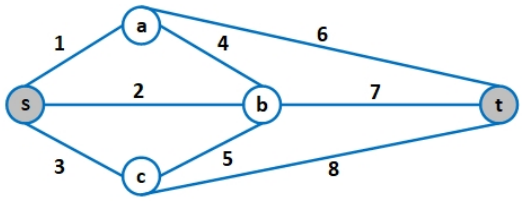


Figure 5.6.2: An instance of the (s,t)-terminal reliability problem

Example 5.6.1 Figure 5.6.2 illustrates an instance of the (s,t)-terminal reliability problem where edges fail (so, $E = \{1,2,3,4,5,6,7,8\}$). Assume we start with a consecutive sequence of cutsets $Q = (c_1 = \{1,2,3\}, c_2 = \{6,7,8\})$, and set $i = 1$. Then we get the following edge sets: $E_{start} = \phi$, $E_{i,i+1} = \phi$, $E_{i,\overline{i+1}} = \{1,2,3\}$, $E_{\bar{i},i+1} = \{6,7,8\}$, $E_{end} = \phi$, and $E_{rem} = \{4,5\}$. ■

Function CS_extend: The pseudo code of function CS_extend is shown in Figure 5.6.3. The main loop of the function (Step 2) performs $|E_{i,\overline{i+1}}| \times |E_{\bar{i},i+1}|$ iterations. In each iteration, we obtain an instance $(G', C, p, \mathbf{P}, k_{req})$ of the EXPO-RU E2C problem (discussed in Chapter 4). Each iteration corresponds to a choice of a pair of nodes (x,y) where $x \in E_{i,\overline{i+1}}$ and $y \in E_{\bar{i},i+1}$ to exclude from a solution X (so, X is guaranteed to be different from c_i and c_{i+1}). The instance is obtained as follows:

- Step 3 constructs G' by assigning the full state to all nodes in $E_{start} \bigcup E_{end} \bigcup \{x,y\}$ from G.
- Step 4 constructs a configuration C by including all nodes in $E_{i,i+1}$.
- Step 5 solves the E2C problem on the formed instance $(G', C, p, \mathbf{P}, k_{req})$.
- In Step 6, if a solution C_{new} exists, set $X = C \bigcup C_{new}$ and return X. Else, the search continues by preforming the next iteration.

Function Q_extend: The pseudo code of function Q_extend is shown in Figure 5.6.4. The function is used to call the above CS_extend function while varying the index i, $i = 1, 2, ..., |Q| - 1$. It returns a potentially larger consecutive sequence Q' of the same type as Q.

Function CS_extend(E, Q, i)
Input: An instance of the EXPO-RU problem $(G, p, \mathbf{P}, k_{req})$, consecutive sequence of cutsets $Q = \{c_1, c_2, ..., c_r\}$ on $|Q| \geq 1$ sets of elements (nodes), and a cutset index i where $i < |Q|$

Output: A set $X \subseteq E$ such that $Q' = (c_1, ..., c_i, X, c_{i+1}, ..., c_r)$ is a consecutive sequence of cutsets.

1. Set $X = \phi$

2. **foreach** (pair (x, y) where $x \in E_{i,\overline{i+1}}$ and $y \in E_{\overline{i},i+1}$)

{ //Form an instance $(G', C, p, \mathbf{P}, k_{req})$ of the E2C problem

3. Construct G' from G by assigning the full state to all nodes in $E_{start} \bigcup E_{end} \bigcup \{x, y\}$
4. Construct a configuration C by assigning a state $s(x)$ to each node $x \in E_{i,i+1}$; the sate $s(x)$ can be either the state of x in cutset c_i or c_{i+1}
5. Solve the instance $(G', C, p, \mathbf{P}, k_{req})$ of the E2C problem

6. **if** (a solution C_{new} exists) **return** $X = C \bigcup C_{new}$
else continue with the next iteration
}

Figure 5.6.3: Pseudo code for function CS_extend

Function Q_extend(Q)
Input: A consecutive sequence Q (of type pathsets or cutsets)
Output: Returns a potentially larger consecutive sequence Q' of the same type as Q

1. Set $Q' = \phi$
2. **for** $(i = 1, 2, ..., |Q| - 1)$
{
3. $X =$ **CS_extend**(E, Q, i)
4. **If** (X exists)
5. Insert X between the sets indexed i and $i + 1$ in Q
}
6. **return** $Q' = Q$

Figure 5.6.4: Pseudo code for function Q_extend

5.6.2 Correctness and Running Time

An important ingredient in the the proof of correctness is the following characterization: for a given index i, $i \in [1, |Q| - 1]$, a cutset X is a solution to an

instance (E, Q, i) of the CS_extend problem **if and only if** it satisfies the two following conditions:

a) X must include all elements in $E_{i,i+1}$, and

b) X can include elements only from $E_{i,\overline{i+1}} \bigcup E_{\bar{i},i+1} \bigcup E_{rem}$, but X must differ from each of c_i and c_{i+1} (e.g., by excluding a node $x \in E(c_i)$ and a node $y \in E(c_{i+1})$).

The following example illustrates the above characterization.

Example 5.6.2 Consider the instance $(E, Q, i = 1)$ of the CS_extend problem in Example 5.6.1. A cutset X is a solution if and only if it satisfies the two conditions:

- X must include elements in $E_{i,i+1} = \phi$
- X can include elements from $E_{i,\overline{i+1}} \bigcup E_{\bar{i},i+1} \bigcup E_{rem} = \{1, 2, 3, 4, 5, 6, 7, 8\}$ but X must differ from c_i and c_{i+1}. Thus, cutset $X = \{2, 3, 4, 6\}$ is a possible solution. ■

Running time: Function CS_extend runs in time $O(|E_{i,\overline{i+1}}|.|E_{\bar{i},i+1}|.n.(n+m))$ on a probabilistic graph G with n nodes and m edges. This follows since each iteration of CS_extend solves an instance of E2C problem by invoking, e.g., the function E2C-BFS, which takes $O(n.(n+m))$. ■

5.7 Numerical Results

In this section, we present numerical results that illustrate the performance and use of our devised algorithms. All algorithms are implemented in Python, and run on a 2.8 GHz laptop with 8 GByte of main memory. We use a 3-state node model (with a set $\{full, reduced, fail\}$ of states), and compare the obtained results with the results obtained in [8] where a more extensive and time consuming iteratively improvable algorithm, called the *factoring* algorithm, is used.

To allow for comparisons, we use the class of double-diagonal grids (x-grids, for short) used in [8]. An x-grid has a structure similar to the graph in Fig. 1.2.1 extended to have any desired number of rows and columns. All nodes have the same full (or reduced) transmission range. A square x-grid of dimension $W \times W$, $W \geq 2$, has its rows (columns) numbered $0, 1, \cdots, W-1$. We place the sink node at the origin ($x = 0, y = 0$), and assume that the intrusion path $\mathbf{P}$ is placed vertically between the rightmost two columns. Only nodes adjacent to $\mathbf{P}$ can sense the path.

Graph legend. We use the following notations:

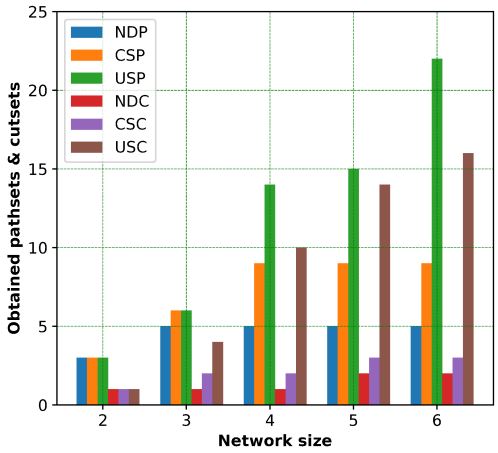
- **NDP (or, NDC)**: node-disjoint pathsets (or, cutsets),
- **CSP (or, CSC)**: consecutive sets of pathsets (or, cutsets), and
- **USP (or, USC)**: unrestricted sets of pathsets (or, cutsets).
- **Factoring**: the factoring algorithm discussed in [20]. We use it to generate a set of pairwise s-disjoint pathset that can be used to obtain bounds from the sum of disjoint products.

5.7.1 Sizes of the Obtained Pathsets and Cutsets

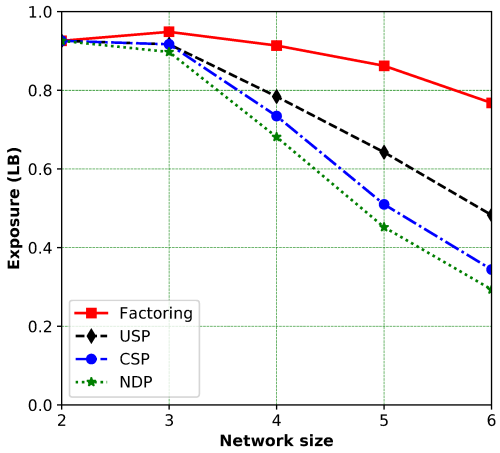
Fig. 5.7.1a shows the number of pathsets and cutsets of each type (node-disjoint, consecutive, and unrestricted) extracted from x-grids of width $W \in [2, 6]$ each when $k_{req} = 1$. As discussed below, the larger number of obtained unrestricted sets results in notable improvements of the bounds. The processing time of any square x-grid using the algorithm ranges from 1.1 msec (millisec) for short sequences (e.g., $r = 5$) to 4.7 msec for longer sequences ($r = 22$). In comparison, performing 1000 iterations of the factoring algorithm on an x-grid of size $W = 6$ requires around one minute.

5.7.2 Comparisons Among Lower Bounds

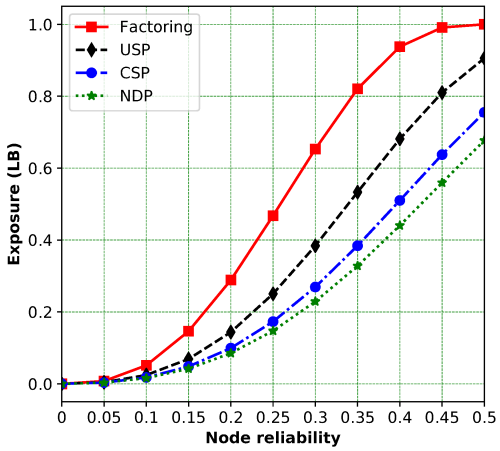
Fig. 5.7.1b illustrates the obtained LBs on x-grids with $W \in [2, 6]$, ($p_{full} = 1/3, p_{red} = 1/3$), and $k_{req} = 1$. As can be seen, the bounds obtained by processing a sequence of unrestricted pathsets improve significantly on the



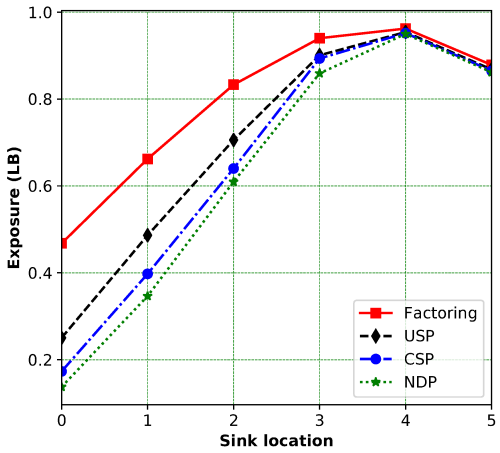
(a) Size of obtained pathsets and cutsets



(b) Exposure versus grid width W



(c) Effect of node state probabilities



(d) Exposure at different sink locations

Figure 5.7.1: The obtained numerical results

NDP and the CSP bounds (thanks to the generality of the algorithm). The factoring algorithm, however, provides better results after performing $N = 1000$ iterations where typically hundreds of pathsets are generated. Similar behaviour has been observed using different node state probabilities, including cases where the probabilities are chosen at random.

5.7.3 Effect of Varying Node State Probabilities

In Fig. 5.7.1c, we use a 6×6 x-grid to explore the quality of $\text{Expo}(G, p)$ obtained when $p_{full} (= p_{red})$ varies in the range $[0.0, 0.5]$, and $k_{req} = 1$. A key feature to explore is the minimum probability, denoted p_{min}, for which

$\text{Expo}(G, p) \geq p_{min}$. Based on the USP curve obtained by the algorithm, one can estimate $p_{min} \approx 0.25$ (and by doing much more work to implement and run the factoring algorithm, one estimates $p_{min} \approx 0.2$). The results are encouraging since designers expect the performance of the overall network to exceed the performance of each single node over a wide range of node operating parameters.

5.7.4 Optimal Sink Placement

In this experiment we aim at tackling a WSN design problem where we want to place the sink at a location that maximizes the exposure measure. Fig. 5.7.1d illustrates the curves obtained when $p_{full} = p_{red} = 0.25$, $k_{req} = 1$, and the sink location is changed diagonally from coordinates $(0, 0)$ to $(5, 5)$ in a 6×6 x-grid.

Both the USP bound obtained by the devised algorithm, and the more extensive factoring algorithm indicate that location $(4, 4)$ is the best location. This location wins for its closeness to the path $\mathbf{P}$, and the quality of the pathsets enabled by this location.

5.8 Concluding Remarks

In this chapter, we use a probabilistic graph model with multistate nodes to model the different energy states caused by fluctuations in the harvested energy in an EH-WSN. We consider two general types of reliability problems, and each type has two problems defined on pathsets and cutsets, respectively. For the cutsets (or, pathsets) version of the first problem type, we devise a dynamic programming approach to compute bounds on the exact solution from sequences of the problem's cutsets (respectively, pathsets). The resulting algorithms are versatile and work for any reliability type problem where the system is coherent and the set of node states form a total order under certain relation. For the cutsets (or, pathsets) version of the second problem type, we are given a disjoint sequence of cutsets (respectively, pathsets), and we extend it to a potentially larger sequence of consecutive cutsets (respectively, pathset) that exploits the multi-state node model.

Chapter 6

Directional Path Exposure with Range Uncertainty

In this chapter, we consider a path exposure problem in EH-WSNs where nodes are equipped with directional communication devices. We formalize a directional path exposure with range uncertainty, denoted DirEXPO-RU, where nodes manage fluctuations in their stored energy by adjusting some of their directional transmission parameters. The DirEXPO-RU problem seeks to quantify the ability of a network to detect and report traversal along a given path. A problem that arises in managing the network resources to maximize this reliability measure is to adjust the transmission beam width of each node, where nodes beam centers are given as input. We formalize a half-width angle selection problem, denoted HWAS, and propose two configuration approaches to configure the transmission beam width of each node in the network. Based on the proposed approaches, we develop a heuristic algorithm to deal with the problem, and the devised algorithm is used in a framework for computing bounds for the DirEXPO-RU problem. Then we compare the obtained results with results obtained using omnidirectional transmission.

Some of the results in this chapter appear in [12].

6.1 Overview of System Model

In this section, we introduce the needed assumptions and notations about the node model, the network model, and the directional transmission model.

6.1.1 Node Operation Model

We adopt the same node model as in previous chapters. Here, however, we assume that each node x in a given EH-WSN is equipped with a directional communication device, and an omnidirectional sensing device. Energy harvested at each node x fluctuates over time. For the purpose of simulating the network (e.g., to obtain the probability distributions mentioned below), we assume that time is divided into equal length slots.

Since wireless transmission is the most energy consuming activity in many WSNs, we assume that such fluctuations affect the transmission range of a node (but does not affect much its sensing range). An energy management unit (EMU) in each node controls a node's state during any time slot. Our work employs a multi-state node model for each node. In its basic form, the model associates 3 states ($full$, $reduced$, and $fail$) with each non-sink node x, defined as explained in previous chapters (e.g., Section 5.1).

6.1.2 Network Reliability Model

We adopt a network reliability model that abstracts the above node dynamics over a long operation time by associating with each node x a probability distribution where for each possible state $s \in \{full, reduced, fail\}$ we know the probability $p_s(x)$ (also denoted $p(x, s)$) that node x is in state s (here, $p_{fail}(x) = 1 - p_{full}(x) - p_{red}(x)$). In addition, we assume that different nodes are assigned different states independently of each other. Such an assumption is common in the literature to simplify the analysis, and also to take care of applications where different nodes perform different tasks independent of each other.

In overall, an EH-WSN is modelled over a long period of time by a *probabilistic graph* (G, p) where $G = (V \cup \{sink\}, E)$ is a directed graph on a set V

of EH wireless nodes, a non-EH and fully operational sink node, and a set E of **directed** communication links (also referred to as directed edges or arcs). The length of each arc $(x, y) \in E$ emanating from a node x depends on the energy state of x and also the directional transmission parameters associated with x, as explained below. We note that G is a directed multigraph with parallel arcs. Parallel arcs from a node a to a node b arise since if (a, b) exists when a is in the full energy state then (a, b) may also exist when a is in the reduced energy state (yet, they are considered two different arcs).

6.1.3 Node Directional Communication Model

We assume that all nodes in G are deployed in a 2-dimensional space, where each node has an (x, y)-coordinate. The transmission beam of each node x (when x is in either the full, or the reduced state) can be obtained from the node's *directionality* parameters, denoted $DIR_{comm}(x)$. Our model uses the following parameters: $DIR_{comm}(x) = \{\Theta_{mid}, \alpha_{full}, \alpha_{red}, R_{full}, R_{red}\}$ where

- Θ_{mid} is a counterclockwise (CCW) angle between two rays emanating from x. The first ray is a horizontal ray (i.e., parallel to the x-axis) that extends to the right. The second ray defines the middle of x's transmission beam, see, e.g., Fig. 6.1.1.

- For $\alpha = \alpha_{full}$, the angle $\Theta_{mid} - \alpha$ (or, $\Theta_{mid} + \alpha$) is a CCW angle between two rays emanating from x. The first ray is a horizontal ray as above. The second ray defines the start (respectively, the end) of x's full range transmission beam (thus, x's beam is of width 2α). When $\alpha = 180°$, the beam is of width $360°$, and node transmission becomes omnidirectional. A similar definition applies when $\alpha = \alpha_{red}$ to describe the reduced transmission beam.

- R_{full} (or, R_{red}) is the maximum x's transmission range when x is in the full (respectively, reduced) state, and the angle α is sufficiently narrow (e.g., $\alpha = 1°$).

- For a given half-width beam angle $\alpha = \alpha_{full}$, the actual full transmission

range of a node x depends on R_{full} and the angle α. We use $R_{full}(\alpha)$ to refer to such an actual transmission range. In general, such a transmission range decreases as α increases. Similarly, we use $R_{red}(\alpha)$ to refer to a node's reduced transmission angle when $\alpha = \alpha_{red}$.

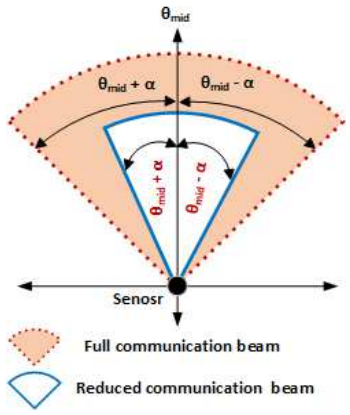


Figure 6.1.1: Node directional communication model.

In Section 6.5, we experiment with grid networks having diagonal links. Each grid network has a sink node located at (x, y)-coordinates $(0, 0)$, and the grid has horizontal (and vertical) links parallel to the x-axis (respectively, the y-axis), in addition to the diagonal links. The horizontal (and vertical) distance between two horizontally (respectively, vertically) adjacent nodes is set to 100 units. For full energy transmission, we set $R_{full}(\alpha = 1^\circ) = 360$ units, and use a function that reduces the transmission range linearly so that $R_{full}(\alpha = 180^\circ) = 180$ units (for omnidirectional transmission). Likewise, for reduced energy transmission, we set $R_{red}(\alpha = 1^\circ) = 180$ units, and $R_{red}(\alpha = 180^\circ) = 100$ units.

6.1.4 The Directional EXPO-RU Problem

In the context of designing EH-WSNs with directional communication nodes, we aim at developing methods to configure node transmission dynamically so as to increase the overall network path exposure reliability. Achieving this goal is non-trivial since the EXPO-RU assessment problem has been shown in [11] to be intractable (#P-hard), and there is no simple optimization criterion that can be used to adjust the node directionality parameters so as maximize

the overall network reliability. We use the name DirEXPO-RU to refer to the EXPO-RU problem when nodes employ directional transmission. We use the concepts of network state, network configuration, and pathset introduced in previous chapters.

Example 6.1.1 Fig. 6.1.2 shows an instance of a 3×3 grid network with one unit of horizontal (or, vertical) spacing. The dashed lines (coloured red) represent communications in full power whereas the solid lines (coloured blue) represent communications in reduced power. In Fig. 6.1.2, $\Theta_{mid} = 135°$, $\alpha_{full} = \alpha_{red} = \frac{135°}{2}$, $R_{full}(\alpha_{full}) = 2$ units, and $R_{red}(\alpha_{red}) = 1$ unit. If $k_{req} = 2$, then the configuration $C = \{(3, reduced), (4, full)\}$ is a pathset and, indeed, it is a minpath.

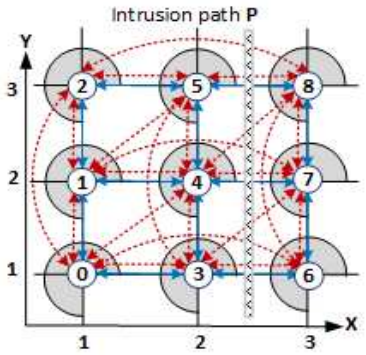


Figure 6.1.2: An instance of a 3×3 grid network.

In Section 6.2, we introduce two approaches for adjusting the directionality parameters so as to obtain good network reliability performance.

6.2 Approaches for Adjusting Directional Transmission

Given a node x in a state $s \in \{full, reduced\}$, we present in this section two approaches for configuring the directionality parameters of the node-state pair (x, s). We assume that the given problem instance specifies (as part of the input) the angle Θ_{mid} that determines the direction of the middle line of x's transmission beam (e.g., one may use the direction of the line segment between

x and the sink node). To adjust the half-width angle $\alpha(x, s)$ for node x in state s, we experiment with the following two approaches that aim at maximizing the performance measure $Expo(G, p)$.

6.2.1 Approach 1

In this approach, we adjust the angle $\alpha(x, s)$ so as to maximize the out-degree, denoted $deg^+(x, s)$, of node x. Equivalently, we seek to maximize the number of out-neighbours reachable from node x in state s. We recall that the actual transmission range $R_{full}(\alpha)$ (or, $R_{red}(\alpha)$) decreases as α increases. The rationale of this approach is that the more directed links that exist in any network state S of the resulting probabilistic graph (G, p), the more likely that $Expo(G, p)$ increases.

We next remark that maximizing $deg^+(x, s)$ for the node-state pair (x, s) is a local operation to node x that does not depend on the directionality setting of other nodes. Our implementation of this approach (present in next section) performs sequential search for finding an optimized angle $\alpha(x, s)$ in the interval $[1°, 180°]$ in increments of some sufficiently small angle δ (e.g., $\delta = 1°$).

6.2.2 Approach 2

The above approach seeks to maximize the number of out-neighbours of node x in state s. In this approach, we consider the quality of such out-neighbours. In particular, this approach makes an effort to adjust the angle $\alpha(x, s)$ so as to give preference to include out-neighbour y of x depending on the quality of the best found directed path, denoted $P^*_{y,sink}$, from y to the sink node. Based on the quality of such a best found path, we associate with node y a preference weight, denoted $w(y)$, that takes on a value that is proportional to the goodness of the corresponding best found directed path $P^*_{y,sink}$.

We next remark that unlike the first approach, finding such a best path $P^*_{y,sink}$ depends on the directionality setting of nodes other than node x (including node y). To simplify the search for such a best path, we adopt a heuristic solution that assumes that each node z, $z \neq x$, operates in an omnidirectional (full or reduced) mode.

As done in approach 1, we perform a sequential search for a good setting of the angle $\alpha(x, s)$ in the range $[1°, 180°]$ in increments of some sufficiently small angle δ (e.g., $\delta = 1°$). At each search step, the angle $\alpha(x, s)$ is assigned a certain value, and each node $z \in V$, $z \neq x$, is assumed to be omnidirectional. With the angle $\alpha(x, s)$ assigned a specific value in each search step, node x can reach a subset of its possible out-neighbours, denoted $S_{\alpha(x,s)}$. With each possible out-neighbour y of x assigned a preference weight $w(y)$, we define the weight of the set $S_{\alpha(x,s)}$ to be $w(S_{\alpha(x,s)}) = \sum_{y \in S_{\alpha(x,s)}} w(y)$. Then, the search algorithm selects an angle $\alpha(x, s)$ that gives the highest $w(S_{\alpha(x,s)})$.

The details of computing the preference weight $w(y)$ of a possible out-neighbour y of x follows the following steps.

1. Let (a, b) be an arc in the probabilistic graph (G, p) that exists when node a is in state $s \in \{full, reduced\}$. We recall that typically the graph G has many parallel arcs since an arc (a, b) that exists when node a is in the full state may also exist when a is in the reduced state (and the two arcs are considered to be different). We associate with each such an arc (a, b) a cost, denoted $cost(a, b)$, that takes on a small value when the node state probability $p(a, s)$ takes on a high value (e.g., $cost(a, b) = -\log p(a, s)$).

2. The cost of a directed path $P_{(y,sink)}$ from a node y to the sink, denoted $cost(P_{(y,sink)})$, is the sum of the costs of its arcs.

3. We set the directed graph G_x to be G with node x deleted. Taking arc costs as distances in G_x and the sink node as a destination, we solve a single-destination shortest paths problem to find a shortest path from each potential out-neighbour y of x to the sink. We denote such a shortest path by $P^*_{y,sink}$.

4. Finally, we assign a preference value $w(y)$ to node y so that the value is inversely proportional to the $cost(P^*_{y,sink})$ (e.g., $w(y) = 1/cost(P^*_{y,sink})$).

We conclude this section by noting that the two above approaches also apply when each node has multiple states (i.e., not only the basic 3-state model used to explain the approaches).

6.3 More Algorithmic Details

In this section, we present a heuristic algorithm, called HWAS (for half-width angle selection), that utilizes the two proposed approaches (explained in Section 6.2) to configure the transmission beams of each individual node x in the network given its working direction Θ_{mid}. The devised HWAS algorithm produces two different types of results based on the selected approach.

6.3.1 Algorithm Idea

Consider an instance (G, p) of the DirEXPO-RU problem with directional parameters $\text{DIR}_{\text{comm}}(x) = \{\Theta_{mid}, \alpha_{full}, \alpha_{red}, R_{full}, R_{red}\}$ for each node $x \in G$, where Θ_{mid}, R_{full}, and R_{red} are given. Our devised algorithm configures the directionality parameters for each individual node $x \in G$ by finding the half-width angle $\alpha(x, s)$ for node x in state s using *Approach 1* and *Approach 2*.

The algorithm utilizes an associative array, denoted $Best_\alpha$, to store such $\alpha(x, s)$ values. In other words, $Best_\alpha$ provides a key-value mapping, where a key is a tuple (x, s) of node x in state s (e.g., $s \in \{full, reduced\}$), and the corresponding value is $\alpha(x, s)$.

In particular, the algorithm configures the directionality parameters of the node-state pair (x, s) by performing the following steps:

1. Perform a sequential search for a good setting of the angle $\alpha(x, s)$ in the range $[1°, 180°]$ in increments of some sufficiently small angle δ (e.g., $\delta = 1°$), which implies adjusting the transmission beam of node x in state s to reach a subset of its possible out-neighbours, denoted $S_{\alpha(x,s)}$. After processing all $\alpha(x, s)$, we obtain a set $\{S_{\alpha_1(x,s)}, ..., S_{\alpha_{180}(x,s)}\}$ of possible out-neighbours of x that correspond to values of $\alpha(x, s)$.

2. We evaluate the obtained set, and select a member $S_{\alpha(x,s)}$ using either **Method-1** or **Method-2** (explained below).

3. Then, we set $Best_\alpha(x, s) = \alpha(x, s)$ that corresponds to the best subset $S_{\alpha(x,s)}$ of out-neighbours of x in state s.

6.3.1.1 Method-1

This method is based on *Approach 1* (explained in Section 6.2). It is a baseline method that selects $\alpha(x, s)$ for each node $x \in G$ based on the number of out-neighbours of x. The approach aims at increasing the reachablity of node x to maximize the performance measure Expo(G, p). In particular, Method-1 works by selecting a member $S_{\alpha(x,s)}$ of $\{S_{\alpha_1(x,s)}, ..., S_{\alpha_{180}(x,s)}\}$ that has the highest number of out-neighbours of x. We set $Best_\alpha(x, s) = \alpha(x, s)$ corresponding to the selected member $S_{\alpha(x,s)}$.

6.3.1.2 Method-2

This method is based on *Approach 2*. It is a cost-based method that selects a subset $S_{\alpha(x,s)}$ for each node $x \in G$ based on *good out-neighbours* of x in state s, and hence can improve the performance measure Expo(G, p). In particular, Method-2 works by selecting a member $S_{\alpha(x,s)}$ of the set $\{S_{\alpha_1(x,s)}, ..., S_{\alpha_{180}(x,s)}\}$ that has the highest weight, $w(S_{\alpha(x,s)})$, as explained in Section 6.2. Then, we set $Best_\alpha(x, s) = \alpha(x, s)$ corresponding to the selected member $S_{\alpha(x,s)}$ of out-neighbours of x in state s.

After computing α_{full} and α_{red} for each node $x \in G$, the actual $full$ (respectively, *reduced*) transmission range of a node x is calculated as explained in Section 6.2. Therefore, $full$ (respectively, *reduced*) arcs for each node $x \in G$ are added based on node's directionality parameters $\text{DIR}_{\text{comm}}(x)$.

6.3.2 Algorithm Details

Fig. 6.3.1 shows a pseudo code for function HWAS. Step 1 initializes array $Best_\alpha$ used to store $\alpha(x, s)$ for all nodes in G. The nested loop in Steps 2 and 3 iterates over each node-state pair (x, s) to find its half-width angle $\alpha(x, s)$. Steps 4 and 5 perform a sequential search for a good setting of an angle $\alpha(x, s)$ in the range $[1°, 180°]$ (in increment of $\delta = 1°$). This process generates a set $\{S_{\alpha_1(x,s)}, ..., S_{\alpha_{180}(x,s)}\}$ of possible subsets for node x's neighbours. Step 6 evaluates each generated set, and selects a subset $S_{\alpha(x,s)}$ using either **Method-1**

or **Method-2**. Step 7 sets $Best_\alpha(x,s) = \alpha(x,s)$ corresponding to the best subset $S_{\alpha(x,s)}$ of the out-neighbours of x in state s. Finally Step 9 returns $Best_\alpha$.

Function HWAS(G,p)
Input: An instance (G,p) of the DirEXPO-RU problem, and $\text{DIR}_{\text{comm}}(x)$ for each node $x \in G$
Output: Compute $\alpha(x,s)$ where $x \in G$ and $s \in \{full, reduced\}$

1. set $Best_\alpha = \{\}$
2. **foreach** (node $x \in G$)
   {
3. **foreach** (node state $s \in \{full, reduced\}$)
   {
4. **for** ($\alpha = 1^\circ$ *to* 180° **step** δ)
   {
5. find set $\{S_{\alpha_1(x,s)}, ..., S_{\alpha_{180}(x,s)}\}$ of subsets of out-neighbours of x as explained in the text
   }
6. evaluate the obtained set and select a subset $S_{\alpha(x,s)}$ for x in state s using **Method-1** or **Method-2** as explained in the text
7. set $Best_\alpha[(x,s)] = \alpha(x,s)$ that corresponds to $S_{\alpha(x,s)}$
   }
   }
8. **return** $Best_\alpha$

Figure 6.3.1: Function HWAS for the DirEXPO-RU problem

6.3.3 Running Time

Consider an input problem instance on a 3-state network with n nodes. Denote by d_{max} the maximum possible out-degree of any node ($d_{max} \leq n-1$). Thus, the maximum possible number of arcs in the network is $m \leq d_{max}(n-1)$.

Method-1 iterates a fixed number of times (depending on the increment angle δ) over every node-state pair (x,s). Each iteration examines at most d_{max} possible out-neighbours of x. Thus, the algorithm requires $O(n \cdot d_{max})$ time.

Similarly, Method-2 iterates over every node-state pair (x,s). Assume a

fixed number of node states (in Fig.6.3.1, we use only 2 states), and assume a fixed number of times Step 4 iterates (depending on the increment δ), and noting that in each iteration of Step 4 one can identify each set $S_{\alpha_i(x,s)}$ of out neighbours of x in $O(n)$ time. We conclude that the algorithm requires $O(n \cdot (n+m))$ time. ■

6.4 Computing Bounds

One main contribution of this work is obtaining lower bounds (LBs) on Expo(G, p, k_{req}) for the DirEXPO-RU problem where each input graph G is constructed by our devised HWAS configuration methods. To compute the Expo measure for any given problem instance, we use an iteratively improvable method, called the factoring method. Reference [7] is among the early references to this general method. Subsequently the method has been used to obtain lower and upper bounds on many reliability problems, including the class of path exposure problems (see, e.g., [8, 20]). In [20], the authors adapt the factoring method to compute lower bounds (LBs) and upper bounds (UBs) of the EXPO problem by generating s-disjoint pathsets and cutsets, respectively.

In more details, the factoring algorithm systematically generates a set of pairwise statistical disjoint (**s-disjoint**, for short) configurations that can be used to obtain bounds from the sum of disjoint products. We call two configurations C_1 and C_2 s-disjoint if at least one node, say x, that appears in both C_1 and C_2 is assigned two different states in the two configurations. So, $\Pr(C_i) + \Pr(C_j)$ is the probability of obtaining at least C_i and/or C_j. For the DirEXPO-RU problem, e.g., the configurations $C_1 = \{(1, reduced), (2, fail)\}$, and $C_2 = \{(1, full), (2, fail)\}$ are s-disjoint since node 1 is assigned two different states in C_1 and C_2.

For the EXPO-RU problem (with omnidirectional transmission), the work in Chapter 3 develops function E2P (for extension to a pathset) that extends (if possible) a given configuration C to a pathset. The E2P function is used within the factoring method to obtain LBs on the solutions. In this context,

the method generates a set $\{P_1, P_2, \cdots, P_r\}$ of pathsets such that

$$\text{Expo}(G, p, k_{req}) \geq \sum_{i=1}^{r} Pr(P_i) \tag{6.4.1}$$

In the next section, we present results based on using this latter method to compute LBs on problem instances generated by our HWAS configuration methods.

6.5 Numerical Results

In this section, we present results to evaluate and compare the performance of the devised directional transmission configuration approaches.

Test networks. The results are for a class of networks that can be viewed as extended 2-dimensional square grid networks (denoted x-grids). Any such $W \times W$ network G has W rows (and columns) indexed as $0, 1, 2, \cdots, W-1$ from bottom to top (respectively, left to right). Each node has (x, y)-coordinates. The sink node is placed at the origin at coordinates $(0, 0)$. Rows (respectively, columns) run horizontally (respectively, vertically) parallel to the x-axis (respectively, the y-axis). The horizontal (or vertical) distance between two consecutive nodes in the grid is set to 100 units.

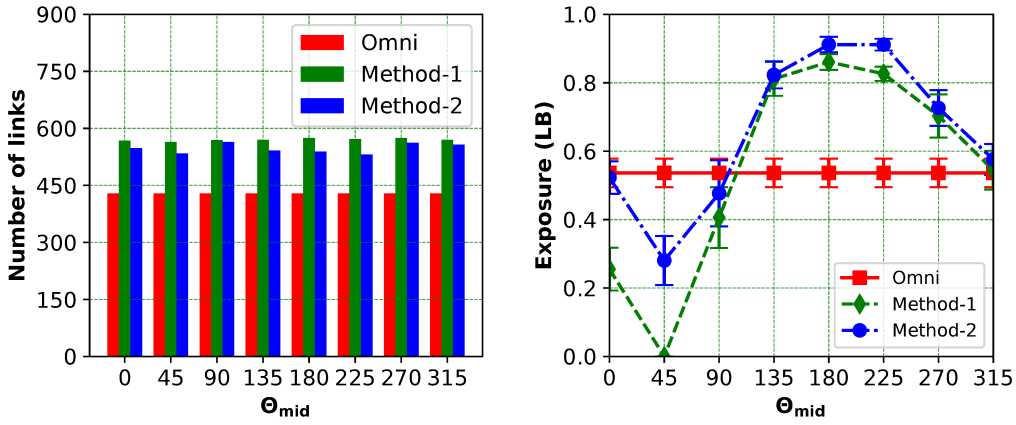
As explained in Section 6.1.3, when a node is in the full energy state its actual transmission range is assumed (for simplicity of obtaining numerical results) to decrease linearly from $R_{full}(\alpha = 1^\circ) = 360$ units to $R_{full}(\alpha = 180^\circ) = 180$ units (the omnidirectional case) as the half-width beam angle α increases. Thus, e.g., with omnidirectional transmission of an internal node x, the node can reach 8 other nodes (corresponding to 4 horizontal and vertical neighbours, and 4 diagonal nodes). Likewise, when a node is in the reduced energy state its actual transmission range is assumed (again, for simplicity) to decrease linearly from $R_{red}(\alpha = 1^\circ) = 180$ units to $R_{red}(\alpha = 180^\circ) = 100$ units as the half-width beam angle α increases. Thus, with omnidirectional transmission of an internal node x, the node can reach 4 other nodes. In each x-grid, the intrusion path $\mathbf{P}$ runs vertically between the rightmost two columns. Only nodes that lie on the immediate left and right of $\mathbf{P}$ sense the

path.

Node-state probabilities. We obtain results with 95% confidence and each point in each obtained curve is the average of 20 runs, where each run assigns to each node x random $p_{full}(x)$ and $p_{red}(x)$ such that $p_{full}(x)+p_{red}(x)+p_{fail}(x) = 1$ and each run completes within 71 seconds.

Reliability lower bounding method. In each run, a LB on Expo(G, p) is obtained by the factoring algorithm after performing 1000 iterations.

Graph legend. Labels *Method-1* and *Method-2* refer to results obtained using our devised HWAS algorithm based on *Approach 1* and *Approach 2* , respectively. The *Omni* label refer to results obtained using omnidirectional transmission.



(a) Number of links

(b) Exposure

Figure 6.5.1: Links and exposure versus $\Theta_{mid} \in [0°, 360°]$

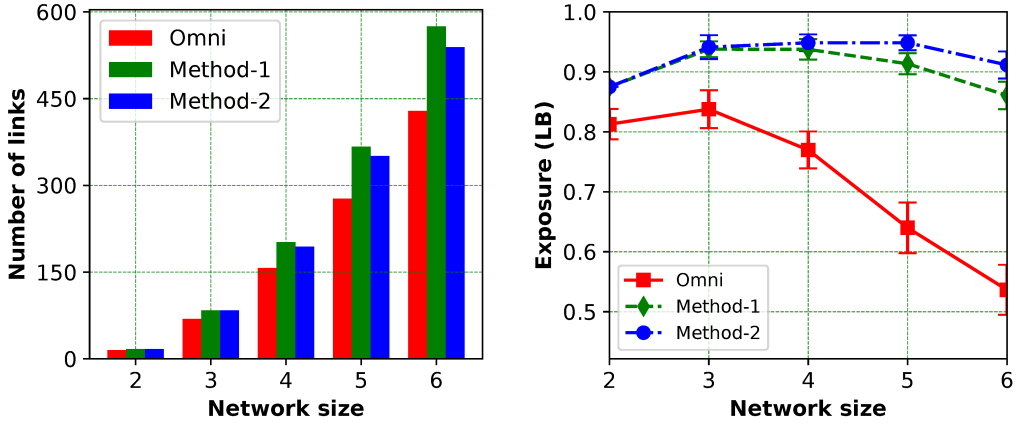
6.5.1 Effect of Varying Θ_{mid}

In this set of experiments, we explore the ability of our two devised approaches to configure the beam-widths of each node so as to achieve good LBs on the exposure reliability measure. We utilize a 6×6 x-grid, and vary Θ_{mid} in the

range $[0°, 360°]$ (all nodes use the same Θ_{mid} value). We recall that for each node x and state $s \in \{full, reduced\}$, the proposed approaches are used to configure the angles $\alpha(x, s)$, and consequently the out-neighbours of node x.

Fig. 6.5.1a presents a histogram of the average number of links in the network that results from using random node-state probabilities (explained above). The average number of links achieved by the omnidirectional configuration is constant (independent of Θ_{mid}). Method-1 has a particular strength in maximizing the out-degree of each node. Thus, it achieves the highest average number of total links in the network. Method-2, however, produces a comparable number of links in the graph.

Fig. 6.5.1b presents curves of the average LB on the computed $\text{Expo}(G, p)$ measure. Method-2 has a particular strength in selecting for each node x reachability to out-neighbours that are deemed to be good in reaching the sink node. Thus, one expects the resulting LBs to outperform LBs obtained from the two other methods. This expected behaviour is confirmed in the figure.



(a) Number of links

(b) Exposure

Figure 6.5.2: Links and exposure versus network size

6.5.2 Effect of Varying Network Size

Here, we show the effects of using different networks of size $W \times W$, where $W \in [2, 6]$ and $\Theta_{mid} = 180°$, on the obtained number of links and exposure.

The proposed approaches are used to configure the transmission beams of each node for both the *full* and *reduced* power states. Fig. 6.5.2a and Fig. 6.5.2b show the obtained number of links and exposure, respectively, for all methods: Method-1, Method-2, and the *Omni* method.

Fig. 6.5.2a shows that the number of links increases as network's size increases for all used methods as a large network will have a higher number of nodes, and hence more links. However, Method-1 obtains the highest number of links compared to Method-2 and the *Omni* method.

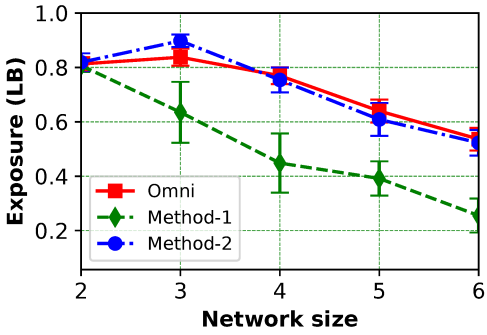
Fig. 6.5.2b shows that the obtained results by Method-1 and Method-2 outperform the *Omni* method for different network sizes. The results illustrate the advantages of using directional sensors over omnidirectional as they provide a higher level of tunability needed in optimizing their performance when network's size increases.

6.5.3 Identifying Good Working Direction Θ_{mid}

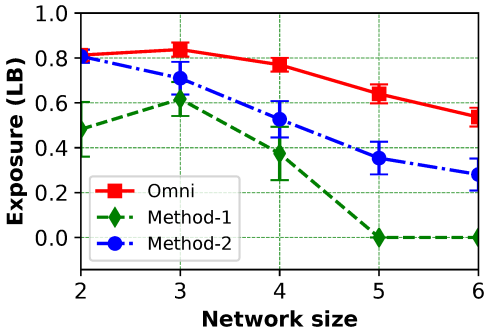
Configuring the direction of the transmission beam center (determined by the angle Θ_{mid}) of nodes in the network is critical for obtaining good performance. In cases when this parameter is misconfigured for a node, it is hoped that by adapting the beam width the node can still deliver acceptable performance.

In this set of experiments, we examine this aspect when using $W \times W$ x-grids and varying W in the range $[2, 6]$. Since the sink node is placed at the (x, y)-coordinate $(0, 0)$, the direction of the line between a node x and the sink varies in the range $[180°, 270°]$ ($180°$ for nodes on the x-axis, and $270°$ for nodes on the y-axis).

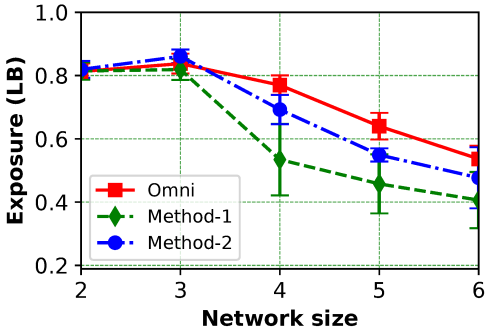
Fig. 6.5.3 shows detailed obtained results when changing Θ_{mid} in the range $[0°, 360°]$. The results show that directional settings outperform the omnidirectional setting when all nodes are oriented so that $\Theta_{mid} \in [180°, 270°]$. The results also show that even when $\Theta_{mid} = 90°$ (a setting that can be viewed as a misconfiguration, given the sink position), the working of Method-1 and Method-2 have been able to adjust the angle α of each node and the obtained LBs are comparable with the omnidirectional case. The results also point to the importance of adjusting the beam-width 2α based on the quality of the



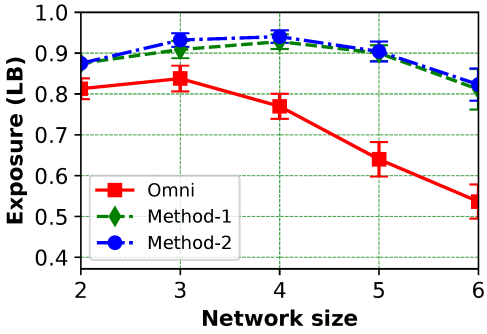
(a) $\Theta_{mid} = 0°$



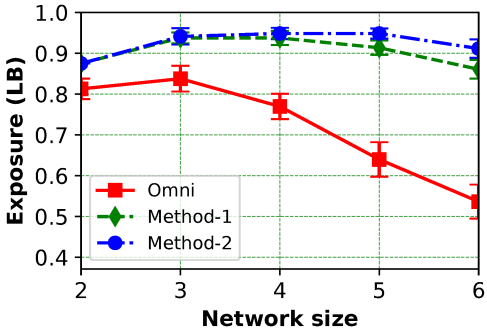
(b) $\Theta_{mid} = 45°$



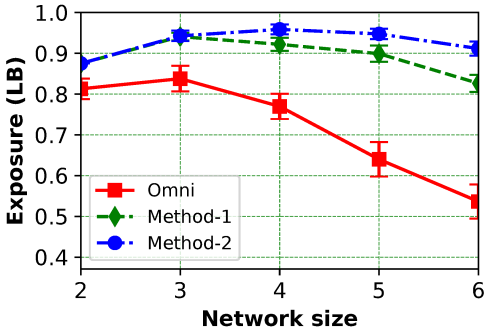
(c) $\Theta_{mid} = 90°$



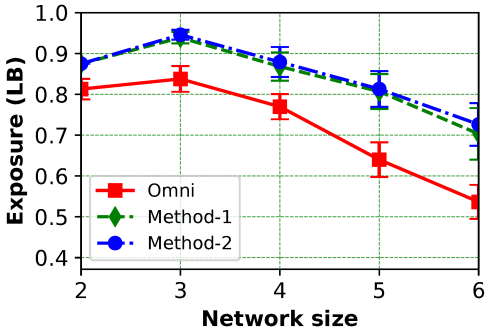
(d) $\Theta_{mid} = 135°$



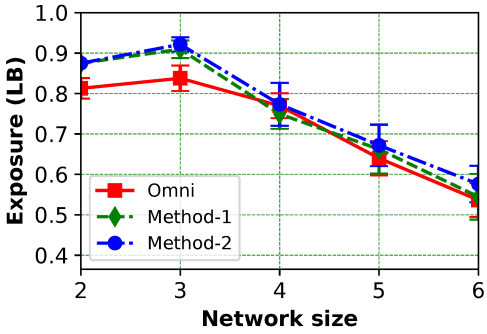
(e) $\Theta_{mid} = 180°$



(f) $\Theta_{mid} = 225°$



(g) $\Theta_{mid} = 270°$



(h) $\Theta_{mid} = 315°$

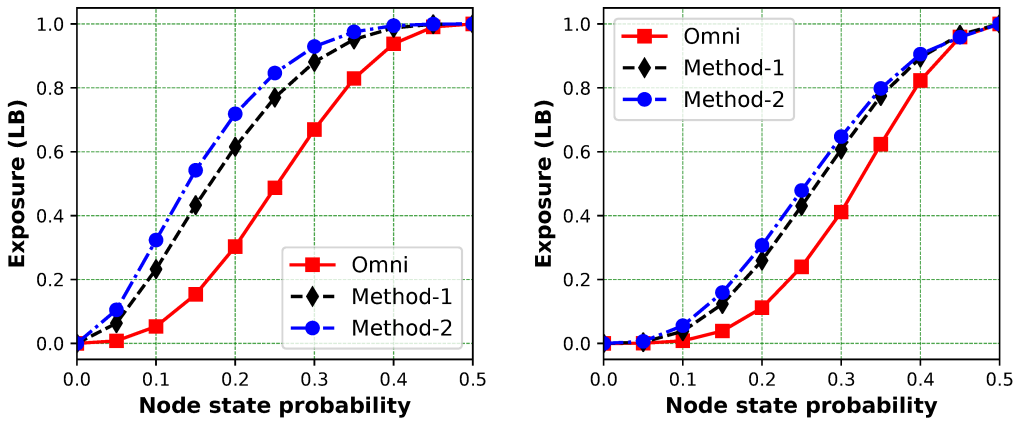
Figure 6.5.3: Obtained bounds versus network size

obtained routes to the sink (as considered in Approach 2).

6.5.4 Exposure Versus Node State Probability

Here, we compare directional transmission with omnidirectional transmission as we set $p_{full}(x) = p_{red}(x) = p$ for each node x, and vary p in the range $[0.0, 0.5]$. The probability p here can be viewed as the node's operating (either in the *full* or *reduced* sates) probability. We note that low p values correspond to cases where the fraction of time when nodes in the network operate in the full or reduced energy states is small. This can happen, e.g., because nodes can not harvest enough energy, or nodes decide to conserve power.

The experiments use a 6×6 x-grid with $k_{req} = 1$ (Fig. 6.5.4a), and $k_{req} = 3$ (Fig. 6.5.4b). In all cases, for any value of p, directional transmission achieves higher average LB on Expo(G, p) than omnidirectional transmission. The results show the advantage of utilizing and properly configuring directional EH-WSNs. The curves also show that directional networks are capable of having an exposure reliability that exceeds the operating probability of any single node in the network.



(a) $k_{req} = 1$ and $\Theta_{mid} = 180°$ (b) $k_{req} = 3$ and $\Theta_{mid} = 180°$

Figure 6.5.4: Exposure versus node state probability

6.6 Concluding Remarks

In this chapter, we consider a fundamental problem on configuring the transmission beams of nodes in a WSN that employs energy harvesting (EH) to achieve prolonged operating time. We take the overall network reliability for a path exposure problem as an objective function that we seek to maximize. We propose two configuration approaches to configure the transmission beams of each node in the network given its transmission beam centre. The proposed approaches have shown the advantages of using directional transmission over omnidirectional transmission.

Chapter 7

Conclusion and Future Work

7.1 Summary

In this thesis, we consider EH-WSNs where fluctuations in a node's stored energy affect primarily its transmission range. We analyze a class of intrusion detection problems where an unauthorized traversal aims at crossing a geographical area guarded by an EH-WSN. The main objective is to develop methodologies to quantify the likelihood that an EH-WSN whose nodes are subject to energy fluctuations can provide simultaneous detection and reporting of unauthorized traversal along a given path to a sink node.

We formalize an EH-WSN reliability problem, called path exposure with range uncertainty (EXPO-RU) that calls for computing the probability that the collaborative work of all nodes in a given EH-WSN succeeds in detecting and reporting an intrusion along a given path. We show that the problem is computationally intractable, and develop iteratively and non-iteratively improvable algorithms for driving lower and upper bounds on the exact solutions of the problem.

In Chapter 3, we formulate the EXPO-RU problem and show that the problem is #P-hard. In addition, we consider the extension to a pathset (E2P) problem for the EXPO-RU problem, and propose an efficient heuristic algorithm to solve the E2P problem. The devised algorithm is used to design both iteratively and non-iteratively improvable LB algorithms for the EXPO-RU problem. In Chapter 4, we extend the work in Chapter 3 by considering the extension to a cutset problem (E2C) for the EXPO-RU problem. We present

two efficient heuristic algorithms to solve the E2C problem, and use the devised algorithms to design both iteratively and non-iteratively improvable UB algorithms for the EXPO-RU problem.

In Chapter 5, we consider two general types of reliability problems on probabilistic graphs where each node can be independently in one of a possible number of node states (such as the EXPO-RU problem). Each type has two problems defined on pathsets and cutsets, respectively. For the cutsets (or, pathsets) version of the first problem type, we devise a dynamic programming approach to compute bounds on the exact solution from sequences of the problem's cutsets (respectively, pathsets). The resulting algorithms are versatile and work for any reliability type problem where the system is coherent and the set of node states form a total order under certain relation. For the cutsets (or, pathsets) version of the second problem type, we are given a disjoint sequence of cutsets (respectively, pathsets), and we extend it to a potentially larger sequence of consecutive cutsets (respectively, pathset) that exploits the multi-state node model.

In Chapter 6, we consider a variant of the EXPO-RU problem where nodes are equipped with directional communication devices, and study the DirEXPO-RU problem. In this context, we consider a fundamental problem on configuring the transmission beams of nodes in a WSN that employs energy harvesting (EH) to achieve prolonged operating time. We take the overall network reliability for the DirEXPO-RU problem as an objective function that we seek to maximize. We propose two configuration approaches to configure the transmission beams of each node in the network given its transmission beam centre. The proposed approaches have shown the advantages of using directional transmission over omnidirectional transmission.

Contributions of thesis based on the work in the thesis appear in publications [8–12].

7.2 Future Work

In this section, we outline some possible future research problems and directions related to the main thrust of the thesis as follows.

- In Chapters 3 to 6, we consider outdoor WSNs where nodes operate on energy harvested from ambient environment (e.g., solar energy). We investigate the well-known path exposure problem in WSNs under range uncertainty. To cope with range uncertainty of nodes in such EH-WSN, we assume that such fluctuations in a node's energy levels affect only the transmission range of the node and we assume that each node has an energy management unit (EMU) that controls node's state as well as its corresponding transmission range during any time slot. Our assumption arises since wireless transmission is the most energy consuming activity in many WSNs. It is worthwhile to extend the work by considering additional effects of energy fluctuations (e.g., effect on both communication and sensing rages).

- In the thesis, we devise algorithms that can work for any given number of N_{state} of node states. We obtain numerical results when $N_{state} = 3$. Obtaining numerical results when $N_{state} > 3$ is expected to help in exploring the trade-off between increasing the model complexity and obtaining more refined results.

- In light of the #P-hardness result of the EXPO-RU problem, our work in the thesis has considered developing efficient algorithms for computing lower and upper bounds on the exact solutions of the problem. I propose to investigate the existence of polynomial time algorithm to solve the problem on useful special classes of networks (e.g., the class of $W \times L$ x-grids with fixed width W).

- In Chapter 6, for the DirEXPO-RU problem, we investigate a fundamental problem on configuring the transmission beams of nodes where nodes beam centers are assumed to be given as input. The obtained results

have shown the advantages of using directional transmission over omni-directional transmission. I propose to extend this work by developing a more comprehensive dynamic configuration algorithm that does not take the beam centers as input.

- Our results in the thesis concern the case where an EH-WSN has single sink. I propose to consider cases where the network has two, or more, sink nodes and develop such algorithms to handle such networks.

References

[1] S. Abougamila, M. Elmorsy, and E. S. Elmallah, "A graph theoretic approach to localization under uncertainty," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7.

[2] ——, "On probabilistic connected components in underwater sensor networks," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, 2018, pp. 200–207.

[3] K. Adu-Manu, N. Adam, C. Tapparello, H. Ayatollahi, and W. Heinzelman, "Energy-harvesting wireless sensor networks (eh-wsns): A review," *ACM Transactions on Sensor Networks*, vol. 14, pp. 1–50, Apr. 2018.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[5] G. Anastasi, M. Conti, M. D. Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009, ISSN: 1570-8705.

[6] J. Bai, M. Fan, J. Yang, Y. Sun, and C. Phillips, "Smart energy harvesting routing protocol for wsn based e-health systems," in *Proceedings of the 2015 Workshop on Pervasive Wireless Healthcare*, ser. MobileHealth '15, Hangzhou, China: ACM, 2015, pp. 23–28.

[7] M. O. Ball, C. J. Colbourn, and J. Provan, "Network reliability," in *Handbook of Operations Research: Network Models*, M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, Eds., Elsevier North-Holland, 1995, pp. 673–762.

[8] A. Basabaa and E. S. Elmallah, "Bounds on path exposure in energy harvesting wireless sensor networks," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2019.

[9] ——, "Bounding path exposure in energy harvesting wireless sensor networks using pathsets and cutsets," in *The 45th IEEE Conference on Local Computer Networks (LCN)*, Nov. 2020.

[10] ——, "Extension algorithms for path exposure in energy harvesting wireless sensor networks," in *2021 International Conference on Computer Communications and Networks (ICCCN)*, 2021, pp. 1–8. DOI: 10.1109/ICCCN52240.2021.9522270.

[11] ——, "Upper bounds on path exposure in EH-WSNs with variable transmission ranges," in *The 46th IEEE Conference on Local Computer Networks (LCN)*, Oct. 2021.

[12] A. Basabaa. and E. Elmallah., "On configuring directional transmission for path exposure reliability in energy harvesting wireless sensor networks," in *Proceedings of the 11th International Conference on Sensor Networks (SENSORNETS)*, INSTICC, SciTePress, 2022, pp. 60–70.

[13] B. Buchli, D. Aschwanden, and J. Beutel, "Battery state-of-charge approximation for energy harvesting embedded systems," in *European Conference on Wireless Sensor Networks*, Springer, 2013, pp. 179–196.

[14] S. Chaturvedi, *Network Reliability*. John Wiley and Sons, Ltd, 2016.

[15] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. K. Saluja, "Sensor deployment strategy for detection of targets traversing a region," *Mobile Networks and Applications*, vol. 8, pp. 453–461, 4 Aug. 2003.

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

[17] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, third. McGraw Hill, MIT Press, 2009.

[18] M. Elmorsy and E. S. Elmallah, "Guarding an area of interest in sensor grids with unreliable nodes," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 6456–6462.

[19] ——, "Packing of cutsets for a breach path detection problem," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–7.

[20] M. Elmorsy, E. S. Elmallah, and H. M. AboElFotoh, "On path exposure in probabilistic wireless sensor networks," in *The 38th IEEE Conference on Local Computer Networks (LCN)*, 2013.

[21] P. Gong, Q. Xu, and T. M. Chen, "Energy harvesting aware routing protocol for wireless sensor networks," in *2014 9th International Symposium on Communication Systems, Networks Digital Sign (CSNDSP)*, Jul. 2014, pp. 171–176.

[22] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[23] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, 2003.

[24] M. K. Jakobsen, J. Madsen, and M. R. Hansen, "DEHAR: A distributed energy harvesting aware routing algorithm for ad-hoc multi-hop wireless sensor networks," in *2010 IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Jun. 2010, pp. 1–9.

[25] X. Jiang, J. Polastre, and D. Culler, "Perpetual environmentally powered sensor networks," in *Fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005, pp. 463–468.

[26] A. Kailas, M. A. Ingram, and Y. Zhang, "A novel routing metric for environmentally-powered sensors with hybrid energy storage systems," in *2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology*, 2009, pp. 42–46.

[27] A. Kollias and I. Nikolaidis, "Seasonally aware routing for thermoelectric energy harvesting wireless sensor networks," *2015 International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*, pp. 1–11, 2015.

[28] L. Liu, X. Zhang, and H. Ma, "Minimal exposure path algorithms for directional sensor networks," in *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, 2009, pp. 1–6. DOI: 10.1109/GLOCOM.2009.5426228.

[29] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless networks with RF energy harvesting: A contemporary survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 757–789, 2015.

[30] ——, "Wireless charging technologies: Fundamentals, standards, and network applications," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1413–1452, 2016.

[31] G. Martínez, S. Li, and C. Zhou, "Wastage-aware routing in energy-harvesting wireless sensor networks," *IEEE Sensors Journal*, vol. 14, pp. 2967–2974, 2014.

[32] S. Megerian, F. Koushanfar, G. Qu, G. Veltri, and M. Potkonjak, "Exposure in wireless sensor networks: Theory and practical solutions," *Wireless Network*, vol. 8, pp. 443–454, 5 Sep. 2002.

[33] N. Pais, B. K. Cetin, N. Pratas, O. J. Velez, N. R. Prasad, and R. Prasad, "Cost-benefit aware routing protocol for wireless sensor networks with hybrid energy storage system," *Journal of Green Engineering*, vol. 1, no. 2, pp. 189–209, Jan. 2011.

[34] S. Peng and C. P. Low, "Energy neutral clustering for energy harvesting wireless sensors networks," in *2013 19th IEEE International Conference on Networks (ICON)*, Dec. 2013, pp. 1–6.

[35] ——, "Energy neutral routing for energy harvesting wireless sensor networks," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013, pp. 2063–2067.

[36] S. Peng, T. Wang, and C. P. Low, "Energy neutral clustering for energy harvesting wireless sensors networks," *Ad Hoc Networks*, vol. 18, pp. 1–16, 2015.

[37] Y. Peng, Z. Li, W. Zhang, and D. Qiao, "Prolonging sensor network lifetime through wireless charging," in *31st IEEE Real-Time Systems Symposium*, 2010, pp. 129–139.

[38] C. Perkins, E. Belding-Royer, and S. Das, *Rfc3561: Ad hoc on-demand distance vector (aodv) routing*, USA, 2003.

[39] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems," in *Fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005, pp. 457–462.

[40] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer Networks*, vol. 67, pp. 104–122, 2014.

[41] C. Rohner, L. M. Feeney, and P. Gunningberg, "Evaluating battery models in wireless sensor networks," in *International Conference on Wired/Wireless Internet Communication*, Springer, 2013, pp. 29–42.

[42] P. Saha, S. Dey, and M. Khanra, "Modeling and state-of-charge estimation of supercapacitor considering leakage effect," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 1, pp. 350–357, 2020.

[43] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Computing Surveys*, vol. 37, no. 2, pp. 164–194, 2005, ISSN: 0360-0300.

[44] J. G. Shanthikumar, "Reliability of systems with consecutive minimal cutsets," *IEEE Transactions on Reliability*, vol. R-36, no. 5, pp. 546–550, Dec. 1987, ISSN: 0018-9529. DOI: 10.1109/TR.1987.5222467.

[45] ——, "Bounding network-reliability using consecutive minimal cutsets," *IEEE Transactions on Reliability*, vol. 37, no. 1, pp. 45–49, Apr. 1988, ISSN: 0018-9529.

[46] N. Sharma, J. Gummeson, D. Irwin, and P. Shenoy, "Cloudy computing: Leveraging weather forecasts in energy harvesting sensor systems," in *7th IEEE Conference on Sensing, Communication and Networking (SECON)*, 2010, pp. 1–9.

[47] M. H. Shazly, E. Elmallah, and J. Harms, "On breach path detection reliability of wireless sensor grids," in *Computer Communications and Networks (ICCCN)*, 2012.

[48] T. Soyata, L. Copeland, and W. Heinzelman, "RF energy harvesting for embedded systems: A survey of tradeoffs and methodology," *IEEE Circuits and Systems Magazine*, vol. 16, no. 1, pp. 22–57, 2016.

[49] H. J. Strayer and C. J. Colbourn, "Consecutive cuts and paths, and bounds on k-terminal reliability," *Networks*, pp. 165–175, 1995.

[50] H. J. Strayer and C. J. Colbourn, "Consecutive cuts and paths, and bounds on k-terminal reliability," *Networks*, vol. 25, pp. 165–175, 1995.

[51] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Communications Surveys Tutorials*, vol. 13, no. 3, pp. 443–461, 2011.

[52] L. Valiant, "The complexity of computing the permanent," *Theoretical Computer Science*, vol. 8, no. 2, pp. 189–201, 1979.

[53] D. J. Vergados, N. A. Pantazis, and D. D. Vergados, "Energy-efficient route selection strategies for wireless sensor networks," *Mobile Networks and Applications*, vol. 13, no. 3-4, pp. 285–296, 2008, ISSN: 1383-469X.

[54] L. Wang and Y. Xiao, "A survey of energy-efficient scheduling mechanisms in sensor networks," *Mobile Networks and Applications*, vol. 11, no. 5, pp. 723–740, 2006, ISSN: 1383-469X.

[55] Q. Wang and H. Hassanein, "A comparative study of energy-efficient (e2) protocols for wireless sensor networks," in *Sensor Network Protocols.* CRC Press, 2006, pp. 5-1 -5–22.

[56] B. Xiao, Y. Shi, and L. He, "A universal state-of-charge algorithm for batteries," in *Proceedings of the 47th Design Automation Conference*, ser. DAC '10, Anaheim, California: Association for Computing Machinery, 2010, pp. 687–692.

Appendix A

Overview of Some Analytical Models

In this section, we present some analytical methods used in a number of papers. The section is organized around a number of relatively short sections, each section focuses on modelling some particular aspect in one paper (or, two related papers by the same authors). Sections based on the work that appears in a published manuscript share the same variable names. Related sections share the same references in their section titles.

A.1 Energy Model for Super-Capacitor and Rechargeable Battery in [3]

The work presented in [3] considers a WSN with a hybrid energy storage system (HESS) in each node that combines the use of a super-capacitor (SC) and a rechargeable battery (RB). In the model, time is divided into equal time slots. At the beginning of time slot t, node n operates with the harvested energy accumulated in the previous time slot $t-1$. At the end of a time slot t, the energy required to transmit a packet is instantaneously removed from the HESS. By taking into account energy leakage, harvesting rate, and transmission load of data packets, the *energy model for the super-capacitor (SC)* at node n is calculated using equations A.1.1 to A.1.3.

Energy model at the beginning of a time slot t:

$$\underbrace{\widehat{E}_{SC}(n,t)}_{e_1} = min\{\big([\underbrace{E_{SC}(n,t)}_{e_2} - \underbrace{\alpha(t-1)}_{e_3}] + [1 - \underbrace{S_2(n,t)}_{e_4}]\,\underbrace{\gamma_n(t-1)}_{e_5}\big), \underbrace{u_{SC}}_{e_6}\} \tag{A.1.1}$$

where:

$e_1 =$ the residual energy in the SC of node n at the **beginning** of time slot t

$e_2 =$ the residual energy in the SC of node n at the end of time slot t

$e_3 =$ the energy leakage of the SC over time slot t

$e_4 =$ a 0/1 value; the value is set to 0 if the harvested energy is directed to the SC of node n in time slot t. Else, it is set to 1 to allow the RB be charging until it gets full

$e_5 =$ the energy harvested by node n in the previous time slot t

$e_6 =$ the maximum capacity of the SC; the $min\{.,.\}$ function prevents the possibility of exceeding the maximum capacity.

Energy model during a time slot t:

$$\underbrace{\widehat{E}_{load,SC}(n,t,R(j))}_{e_1} = \underbrace{l(j)}_{e_2}\,\underbrace{E(n,R(j))}_{e_3}.\,\underbrace{I[\widehat{E}_{SC}(n,t) - l(j)E(n,R(j)) > 0]}_{e_4} \tag{A.1.2}$$

where:

$e_1 =$ the energy consumed by node n **during** time slot t to relay packet j on route R if the super-capacitor (SC) is used

$e_2 =$ length of packet $R(j)$ in bytes

$e_3 =$ the consumed energy per byte to transmit packet $R(j)$ from node n

$e_4 =$ a 0/1 indicator function I, where $I = 1$ if the residual energy in the SC at the start of time slot $t >$ the amount of energy required to relay the packet $R(j)$. Else, the function $I = 0$

Energy model at the end of a time slot t:

$$\underbrace{E_{SC}(n,t)}_{e_1} = \underbrace{\beta(n,D,t)}_{e_2}[\underbrace{\widehat{E}_{SC}(n,t)}_{e_3} - \underbrace{\widehat{E}_{load,SC}(n,t,R(j))}_{e_4}] \tag{A.1.3}$$

where:

$e_1 =$ the residual energy in the SC of node n at the **end** of time slot t

$e_2 =$ a 0/1 indicator function for the event that the RB of node n has not exceeded its cycle lifetime at the beginning of time slot t

$e_3 =$ the residual energy in the SC of node n at the beginning of time slot t (from equation A.1.1)

$e_4 =$ the energy consumed from the super-capacitor (SC) of node n at time slot t to relay packet $R(j)$ (from equation A.1.2)

The energy model for the rechargeable battery (RB) uses three similar equations (see [3]).

A.2 Routing Based on Cost-benefit Function in [3]

The above equations are used in [3] to compute a cost-benefit function (A.2.1) that reflects the ability of a node n to transmit a data packet at each time slot t. In particular, the lower the cost function, the more able a node to forward a packet.

$$\underbrace{cost(n,t)}_{e_1} = [\underbrace{h_c(n,t) * w_{h_c}}_{e_2} + \underbrace{q_{oc} * w_{q_{oc}}}_{e_3}] - [\underbrace{E_{SC}(n,t) * w_{SC}}_{e_4} + \underbrace{E_{RB}(n,t) * w_{RB}}_{e_5} + \underbrace{\gamma(n,t) * w_{\gamma}}_{e_6} + \underbrace{L_c(n,t) * w_{L_c}}_{e_7}] \tag{A.2.1}$$

where:

$e_1 =$ the cost function (can be negative) for node n at time slot t; this cost function is used to determine the ability of a node to forward a data packet to the sink.

$e_2 =$ the hopcount (the minimum path length) to the sink (weighted by w_{h_c})

$e_3 =$ the length of the node's transmission queue (weighted by w_{qc})

$e_4 =$ the residual energy in the SC of node n at time slot t (weighted by w_{SC})

$e_5 =$ the residual energy in the RB of node n at time slot t (weighted by w_{RB})

$e_6 =$ the energy harvested by node n at time slot t (weighted by w_γ)

$e_7 =$ the cycle lifetime of the RB (weighted by w_{L_c})

Each parameter in the above cost function (e.g., $h_c(n,t)$) is multiplied by a weight and each parameter is set to a value in the range $[0, 100]$ (e.g., $E_{RB}(n,t) = 100$ means the RB is fully charged). Whenever a node has a data packet to route, it queries all of its neighbours about their respective costs to route the packet. The routing algorithm then forwards the packet to the neighbour having the smallest cost.

In the simulation section, the authors set the energy leakage of the SC per time slot ($\alpha(t)$) to $0.1mJ$, the maximum capacity of the RB (u_{RB}) to $75J$, the RB cycle lifetime (L_c) starts with value 1150 sec and decreases 0.5 sec after each discharge. Each node generates data packets of size $L(j) = 24$ bytes with data rate 1 packet/second, the energy spent in transmitting and receiving 1 byte of data is $59.2\mu J$ and $28.6\mu J$, respectively. The duration of each time slot is $T = 100ms$, the energy harvested by a node at time slot t is $\gamma(n,t) = 1mJ$, which corresponds to the energy harvested by an outdoor photo-voltaic cell in a time slot.

A.3 Basic Node Energy Model for Cluster Routing in [4, 5]

In [4, 5], Peng et al. consider EH-WSN with N energy harvesting sensor nodes that are grouped into K clusters and each cluster has a cluster head (CH) that gathers, aggregates data packets received from cluster members (CMs) and sends the data to the base station.

The energy consumed by the transmitter of a sensor node n to transmit one bit of data (denoted, $E_{Tx/bit}(n)$) is given by:

$$E_{Tx/bit}(n) = \begin{cases} \varepsilon_{tx} + \varepsilon_{sp} * d^2 & if\ d < \widehat{d} \\ \varepsilon_{tx} + \varepsilon_{mp} * d^4 & if\ d \geq \widehat{d} \end{cases} \tag{A.3.1}$$

where:

- ε_{tx} is the energy consumed by the transmitter electronics to perform signal filtering, modulation and code spreading.
- ε_{sp} is a constant depending on the required receiver's signal-to-noise ratio (SNR) so that the bit error rate is acceptable assuming the free space path loss model (d^2 power loss).
- ε_{mp} is a constant depending on the required receiver's SNR assuming a two-ray mulitpath propagation model (d^4 power loss).

Similarly, the energy consumed by the *receiver* of a sensor node n to receive one bit of data (denoted, $E_{Rr/bit}(n)$) is estimated as follows:

$$E_{Rx/bit}(n) = \varepsilon_{rx} \tag{A.3.2}$$

where ε_{rx} is the energy consumed by the receiver electrics to receive one bit of data.

Based on equation A.3.1, the total energy consumed by a cluster member (CM) to sense and transmit one bit of data to a cluster head (denoted, $E_{CM}(n)$) is computed as follows:

$$E_{CM}(n) = \varepsilon_{Sx} + E_{Tx/bit}(n) \tag{A.3.3}$$

where ε_{Sx} represents the energy consumed to sense one bit of data.

Similarly, the total energy consumed by a cluster head (CH) to handle (receive, aggregate, transmit) one bit of data sent by a cluster member (denoted, $E_{CH}(n)$) is estimated as follows:

$$E_{CH}(n) = E_{Rx/bit}(n) + \alpha\varepsilon_{da} + \frac{E_{Tx/bit}(n)}{\alpha} \tag{A.3.4}$$

where:

- α is data aggregation factor. If a perfect aggregation method is used, α is set to the number of cluster members inside the cluster. Else, set $\alpha = 1$

- ε_{da} is average energy consumed in data aggregation process to process and aggregate one bit of data

A.4 Node Energy Budget in a Time Slot in [4, 5]

In order to track the harvested solar energy during a typical day (24 hours), the authors in [4, 5] divide the system's time into a number of time slots that are indexed by $t = 1, 2, \ldots, N$. The length of each time slot is denoted T. Thus, the amount of predicted harvested energy during a time slot t by a sensor node n (denoted, $E_H^n(t)$) is given by:

$$E_H^n(t) = \int_a^b P_h(t)dt \tag{A.4.1}$$

where $P_h(t)$ is the predicted harvested power rate in time slot t where a and b are the start and end of the time slot t, respectively.

To maintain the energy neutral state, the energy consumed by a sensor node should not be more than the harvested energy by the sensor during a specific number of time slots, denoted N_t. Let $E_B^n(t)$ denote the amount of energy that node n can use in time slot t (called the energy budget function). So, to preserve the energy neutrality during a period of N_t time slots, we need to satisfy the following expression:

$$\sum_{t=1}^{N_t} E_B^n(t) \leq \sum_{t=1}^{N_t} E_H^n(t) \tag{A.4.2}$$

The authors noted that energy storage devices such as batteries and super-capacitors usually suffer from energy storage inefficiencies. For simplicity, the authors deal with a system with no such inefficiencies, and hence, set $E_B^n(t) = E_H^n(t)$. In the simulation work, the authors set the length of each time slot $T = 1$ hour, and set the harvest energy $E_H^n(t) = 54 + X$ Joules, where X is a random variable drawn from a random distribution $N(0, 4)$.

A.5 Estimating Cluster Head Group Size in [4, 5]

The work in [4, 5] manages an EH-WSN by partitioning the N nodes into K clusters, where K is a user specified number. Each cluster is composed of cluster members (CMs) and nodes in a cluster head group (CHG). At the beginning of each time slot t, nodes spend time forming clusters, selecting a cluster head group, and cluster members for each cluster. The remaining time, denoted T_{dx}, is used to process the sensed information bits. Nodes in a CHG do not perform sensing tasks, rather they process sensed information bits from CMs, aggregate the information, and send the information to the base station. Nodes in a CHG are scheduled so that at any instant one node serves as an active cluster head while the remaining nodes in the CHG go to sleep.

In this section we explain the method used in [4, 5] to estimate the size of a CHG (denoted, N_G^k) for a cluster k, where $1 \leq k \leq K$. To start, the authors observe that since the number of control messages exchanged for cluster formation is small compared with the number of data packets that will be sent in each time slot, the energy spent on exchanging these control message can be neglected. Hence, the maximum amount of data bits in one time slot t (denoted, $B_{max}^k(t)$) that can be transmitted by all cluster members (N_{CM}^k) in cluster k without compromising energy neutrality, can be estimated as follows:

$$B_{max}^k(t) = \sum_{n=1}^{N_{CM}^k} \frac{E_B^n(t)}{E_{CM}^n} \tag{A.5.1}$$

where:

- E_{CM}^n is the amount of energy consumed by cluster member n to sense and transmit one bit of data (from equation A.3.3)
- $E_B^n(t)$ is the amount of energy budget assigned to cluster member n in time slot t (from equation A.4.2)
- $\frac{E_B^n(t)}{E_{CM}^n}$ is the amount of data bits that cluster member n can sense and transmit in a time slot t without compromising its energy neutral state.

Based on equation A.5.1, to ensure energy neutrality of all nodes in the cluster head group (CHG), the following equation should be satisfied:

$$\sum_{n=1}^{N_G^k} \frac{E_B^n(t)}{E_{CH}(n)} \geq B_{max}^k(t) \tag{A.5.2}$$

where:

- $E_{CH}(n)$ is the energy consumed by node n in the CHG to handle (receive, aggregate, transmit) one bit of data (from equation A.3.4)
- $E_B^n(t)$ is the amount of energy budget assigned to node n in the CHG at time slot t (from equation A.4.2)
- N_G^k is the number of nodes in the cluster head group (CHG)
- $\frac{E_B^n(t)}{E_{CH}(n)} (\geq B_{max}^k(t))$ is the maximum amount of data in bits that node n in the CHG can handle without compromising its energy neutrality. Thus, the *LHS* of the inequality (equation A.5.2) denotes the maximum amount of data in bits that all nodes in the cluster head group (CHG) can handle without compromising the energy neutrality of any node in the CHG.

Based on equation A.5.2, as the size of the cluster head group N_G^k increases, the *LHS* increases. However, the *RHS* decreases as the number of cluster members N_{CM}^k decreases. As a result, there will be fewer cluster members (CMs) to gather data to send to the CHG. The size of the cluster head group (CHG) is chosen to be the smallest integer value N_G^k that satisfies inequality A.5.2.

A.6 Node Scheduling in [4, 5]

In this section, we present the key ideas used in scheduling nodes in a typical cluster k. The time period T of a typical time slot can be written as $T = \beta + T_{dx}$, where β is an initial part of T used for cluster formation, and deciding on CHG nodes (and, the CMs) in each cluster k, and T_{dx} is the remaining part of T used in collecting, processing, and relaying information bits.

Scheduling of nodes in a cluster member (CM): In a cluster k with N_{CM}^k member nodes, each node n is scheduled so that it transmits for a total of $\frac{T_{dx}}{N_{CM}^K}$ time during a time slot. Thus, the maximum data transmission rate by any such member n is given by:

$$D_{tx}^n(t) = \frac{B_{max}^n(t)}{\frac{T_{dx}}{N_{CM}^k}} = \frac{E_B^n(t) * N_{CM}^k}{E_{CM}^n * T_{dx}} \tag{A.6.1}$$

Scheduling of nodes in a cluster head group (CHG): In a cluster k with N_G^k nodes in its CHG, the scheme sets a CHG node n to be an active head for a period of time denoted T_A^n. To derive an expression for T_A^n that satisfies energy neutrality, the authors observe that:

- $B_{max}^k(t)$ is the maximum amount of data bits generated by all cluster members in one time slot (equation A.5.1)
- $\frac{B_{max}^k(t)}{T_{dx}}$ is the maximum data rate generated by all cluster members in one time slot
- $\frac{B_{max}^k(t)}{T_{dx}} T_A^n$ is the maximum amount of data handled by cluster head n in one time slot

Thus, the maximum feasible T_A^n time should satisfy:

$$\frac{B_{max}^k(t)}{T_{dx}} * T_A^n = \frac{E_B^n(t)}{E_{CH}^n} \tag{A.6.2}$$

Each node n in the CHG knows $E_B^n(t), E_{CH}^n$, and T_{dx}. In addition, after deciding on the CHG (and the CMs) the node is informed about $B_{max}^k(t)$. Thus, each node n in a CHG can compute its T_A^n time. To verify that $\sum_{n=1}^{N_G^k} T_A^n$ (that is, the CHG can serve during the T_{dx} interval), we check that:

$$\sum_{n=1}^{N_G^k} T_A^n = \frac{\sum_{n=1}^{N_G^k} \frac{E_B^n(t)}{E_{CH}^n}}{\frac{B_{max}^k(t)}{T_{dx}}} \geq T_{dx} \tag{A.6.3}$$

A.7 Overcharge Wastage-Aware Routing in EH-WSNs in [2]

In [2], the authors model EH-WSNs with a graph G where each node $v_i \in G$ has an energy storage unit with capacity B. In their work, time is slotted

around slots of length T_h seconds each ($T_h = 10$ seconds, or 100 seconds in the simulation section). In any time slot, when a node harvests enough energy, any additional received energy is wasted. The work formalizes a route selection problem that aims at reducing such overcharge wastage in each time slot. More details are highlighted below.

Table A.1: Notation

Notation	Description
T_h	A specified prediction time
e_i	Current residual energy at node v_i
e_{h_i}	Estimated harvested energy during T_h
e_{c_i}	Estimated consumed energy during T_h
B	Battery capacity
σ_n	A route between source and sink

Problem formulation: The model uses the following variables (see Table A.1): e_i denotes node v_i energy at the beginning of a time slot, e_{h_i} is the estimated harvested energy during T_h, e_{c_i} is the consumed energy if node v_i is active transmitting and receiving packets during T_h, and σ_n is a route between some source and destination nodes where every node v_i along the route is active during T_h.

In the following equations, $e_{w\_on\_i}$ (respectively, $e_{w\_off\_i}$) denotes the overcharge wastage when node v_i lies on the active route σ_n (respectively, does not lie on σ_n) during T_h.

$$e_{w\_on\_i} = max[0, (e_i + e_{h\_i} - e_{c\_i} - B)] \tag{A.7.1}$$

$$e_{w\_off\_i} = max[0, (e_i + e_{h\_i} - B)] \tag{A.7.2}$$

Thus, the sum of energy consumption due to communication for nodes on the active route σ_n, and the total network wastage of all nodes is given by:

$$C(\sigma_n) = \sum_{v_i \in \sigma_n} (e_{c\_i} + e_{w\_on\_i}) + \sum_{v_i \notin \sigma_n} e_{w\_off\_i} \tag{A.7.3}$$

The devised routing scheme, in [2], aims at finding a route σ_n at each time slot T_h (between given source and destination nodes) that minimizes the cost

$C(\sigma_n)$. The work, in [2], does not give an exact algorithm to find such optimum route σ_n, rather the authors modify the well-known Distance Source Routing (DSR) protocol [1] to find the best possible route from a subset of all possible routes. Roughly speaking,

- The scheme modifies the headers of the Route Request (RREQ) messages to collect information needed to compute $\sum_{v_i \in \sigma_n}(e_{c\_i} + e_{w\_on\_i})$ over a traversed route σ_n.

- The destination waits for sometime to receive a subset of routes from which it can compute a lower bound on $\sum_{v_i \notin \sigma_n} e_{w\_off\_i}$ (the off-route information).

- The destination waiting time is controlled by a timer, called *RREQ* timer (T_{rwq}). The authors remark that T_{rwq} should be small enough to avoid unacceptable route acquisition delay, but large enough to allow possible optimal routes to be received.

- In addition, each selected source-destination route is invalidated after a route expiration time period. This feature allows routes to be updated in response to changes in node energy states.

References of Appendix A

[1] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, T. Imielinski and H. F. Korth, Eds. Boston, MA: Springer US, 1996, pp. 153–181.

[2] G. Martínez, S. Li, and C. Zhou, "Wastage-aware routing in energy-harvesting wireless sensor networks," *IEEE Sensors Journal*, vol. 14, pp. 2967–2974, 2014.

[3] N. Pais, B. K. Cetin, N. Pratas, O. J. Velez, N. R. Prasad, and R. Prasad, "Cost-benefit aware routing protocol for wireless sensor networks with hybrid energy storage system," *Journal of Green Engineering*, vol. 1, no. 2, pp. 189–209, Jan. 2011.

[4] S. Peng and C. P. Low, "Energy neutral clustering for energy harvesting wireless sensors networks," in *2013 19th IEEE International Conference on Networks (ICON)*, Dec. 2013, pp. 1–6.

[5] S. Peng, T. Wang, and C. P. Low, "Energy neutral clustering for energy harvesting wireless sensors networks," *Ad Hoc Networks*, vol. 18, pp. 1–16, 2015.