6-1-2017

Reliable cost-optimal deployment of wireless sensor networks

Dina Salah DeifAllah

Dissertation Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in Electronics and Communications Engineering on

Reliable Cost-Optimal Deployment of Wireless Sensor Networks

Dina Salah Deif Allah

Department of Electronics and Communications Engineering

School of Sciences and Engineering

The American University in Cairo

Spring 2017

Acknowledgements

Firstly, I would like to thank Mr. Yousef Jameel for his generosity in funding the PhD in Applied Sciences and Engineering Fellowship. I am honored to be one of the recipients of this Fellowship. Without it, it would not have been possible for me to conduct the research in this thesis.

I would like to express my sincere gratitude to my thesis advisor, Prof. Yasser Gadallah, for his continuous support and guidance during my Ph.D. journey. The mentorship and encouragement he provided me throughout the time of research and writing of this thesis have been invaluable to me.

I would also like to thank the rest of my thesis committee: Prof. Ayman El Ezabi and Prof. Mohammed Moustafa, for their insightful comments on my dissertation proposal and final dissertation.

My sincere thanks also go to Eng. Mostafa El-Tager who kindly helped me in the experimental part of Chapter 5 of this dissertation.

Last but not least, I would like to thank my family: my parents and my husband for their emotional support and their patience throughout my Ph.D. journey. Their love, understanding and encouragement have helped persevere through the most challenging times.

Abstract

Wireless Sensor Networks (WSNs) technology is currently considered one of the key technologies for realizing the Internet of Things (IoT). Many of the important WSNs applications are critical in nature such that the failure of the WSN to carry out its required tasks can have serious detrimental effects. Consequently, guaranteeing that the WSN functions satisfactorily during its intended mission time, i.e. the WSN is reliable, is one of the fundamental requirements of the network deployment strategy. Achieving this requirement at a minimum deployment cost is particularly important for critical applications in which deployed SNs are equipped with expensive hardware. However, WSN reliability, defined in the traditional sense, especially in conjunction with minimizing the deployment cost, has not been considered as a deployment requirement in existing WSN deployment algorithms to the best of our knowledge. Addressing this major limitation is the central focus of this dissertation. We define the reliable cost-optimal WSN deployment as the one that has minimum deployment cost with a reliability level that meets or exceeds a minimum level specified by the targeted application. We coin the problem of finding such deployments, for a given set of application-specific parameters, the Minimum-Cost Reliability-Constrained Sensor Node Deployment Problem (MCRC-SDP). To accomplish the aim of the dissertation, we propose a novel WSN reliability metric which adopts a more accurate SN model than the model used in the existing metrics. The proposed reliability metric is used to formulate the MCRC-SDP as a constrained combinatorial optimization problem which we prove to be NP-Complete. Two heuristic WSN deployment optimization algorithms are then developed to find high quality solutions for the MCRC-SDP. Finally, we investigate the practical realization of the techniques that we developed as solutions of the MCRC-SDP. For this purpose, we discuss why existing WSN Topology Control Protocols (TCPs) are not suitable for managing such reliable cost-optimal deployments. Accordingly, we propose a practical TCP that is suitable for managing the sleep/active cycles of the redundant SNs in such deployments. Experimental results suggest that the proposed TCP's overhead and network Time To Repair (TTR) are relatively low which demonstrates the applicability of our proposed deployment solution in practice.

Abbreviations

Ant Colony Optimization	ACO
Artificial Potential Field	APF
Branch and Bound	B&B
Branch and Cut	B&C
Breadth-First Search	BFS
Computational Geometry	CG
Delaunay Triangulation	DT
Disjoint Sets Cover	DSC
Fault Tree	FT
Field of View	FoV
Fixed-Length Genetic Algorithm	FLGA
Genetic Algorithm	GA
Greedy Heuristic	GH
Hybrid Genetic Algorithm	HGA
Integer Linear Programming	ILP
Local Search	LS
Maximum-Coverage SDP	MC-SDP
Mean Time To Fail	MTTF
Memetic Algorithm	MA
Minimum Cost Coverage SDP	MCC-SDP
Minimum-Cost Connectivity-Guaranteed SDP	MCCG-SDP
Minimum-Cost Reliability-Constrained SDP	MCRC-SDP
Min-Max Ant System	MMAS
Mixed Integer Programming	MIP
Mobile Wireless Sensor Network	MWSN
Monte Carlo	MC
Multiple Objective Genetic Algorithm	MOGA
Non-deterministic Polynomial time-hard	NP-hard
Non-deterministic Polynomial time-complete	NP-complete
Particle Swarm Optimization	PSO
Point of Interest	PoI

Quality of Service	QoS
Region of Interest	RoI
Reliability Block Diagram	RBD
Reliability-Constrained SDP	RC-SDP
Reliable Deployment TTCP	RD-TTCP
Sensor Node	SN
Sensor Node Deployment Problem	SDP
Swarm Intelligence	SI
Temporal Topology Control Protocol	TTCP
Time To Repair	TTR
Topology Control Protocol	TCP
Virtual Force	VF
Voronoi Diagram	VD
Wireless Sensor Network	WSN

Figures

Tables

Symbols

r_s	SN coverage range
r_c	SN communication range
T_m	WSN mission time
$R_c(t)$	Reliability function of a device
R_c	Reliability of a device during T_m
π_i	Binary state indicator of component i in a system $\boldsymbol{S}$
$\boldsymbol{\pi}$	State of system $\boldsymbol{S}$
$\boldsymbol{\Pi}$	Set of all possible states of system $\boldsymbol{S}$
$f(\boldsymbol{\pi})$	Structure function of system/network $\boldsymbol{S}$
$R(\boldsymbol{S})$	Reliability of system $\boldsymbol{S}$ during T_m
$\boldsymbol{S}_1(\boldsymbol{\pi})$	Set of functional components in system $\boldsymbol{S}$ at state $\boldsymbol{\pi}$
$\boldsymbol{S}_0(\boldsymbol{\pi})$	Set of failed components in system $\boldsymbol{S}$ at state $\boldsymbol{\pi}$
$\boldsymbol{\Pi}_1$	Set of all paths of system $\boldsymbol{S}$
$\boldsymbol{\Pi}_0$	Set of all cuts of system $\boldsymbol{S}$
$\boldsymbol{D} = \{d_i\}, i = 1, \dots, \lvert\boldsymbol{D}\rvert$	Set of possible deployment points in a given RoI of the WSN
$\boldsymbol{T} = \{t_j\}, j = 1, \dots, m$	Set of target points in a given RoI of the WSN
$\boldsymbol{Y}_j$	Set of SNs in the on state that monitor t_j
$\boldsymbol{Z}_j$	Subset of SNs belonging to $\boldsymbol{Y}_j$ which have a functional path to the sink node
λ_t^i	Transceiver probability of failure of SN i during T_m
λ_s^i	Sensor probability of failure of SN i during T_m
$\boldsymbol{X}_t$	Subset of SNs belonging to $\boldsymbol{S}$ with failed transceivers
$\boldsymbol{X}_s$	Subset of SNs belonging to $\boldsymbol{S}$ with failed sensors
$\boldsymbol{X}_t{}^c$	Complement of $\boldsymbol{X}_t$
$\boldsymbol{X}_s{}^c$	Complement of $\boldsymbol{X}_s$
$\boldsymbol{F}$	Set of all the SN failure combinations that can be tolerated by $\boldsymbol{S}$
$Prob(F_q)$	Probability of occurrence of the SN failure combination q in $\boldsymbol{F}$
$R_{lb}(\boldsymbol{S})$	Lower bound of $R(\boldsymbol{S})$
η_{lb}	Lower bound probability threshold
$\lvert\boldsymbol{F}\rvert_{lb}$	number of probability terms include in $R_{lb}(\boldsymbol{S})$ such that $Prob(F_{\lvert\boldsymbol{F}\rvert_{lb}}) \geq \eta_{lb}$
$\boldsymbol{F}_t^k$	set that holds all the tolerable transceiver failure combinations of length k assuming no sensor failures
$\boldsymbol{F}_s^k$	Set of all the tolerable sensor failure combinations of length k assuming no transceiver failures
$\boldsymbol{F}^k$	Set of all the tolerable components failures of length k
x_s, x_t, x_p, x_b	Binary indicators for the state of the sensor, transceiver, processor and battery of a SN respectively
λ_s	Probability of failure of a SN sensor conditioned on the event that the processor and battery are functional during T_m

λ_t	Probability of failure of a SN transceiver conditioned on the event that the processor and battery are functional during T_m
λ_p	Probability of failure of a SN processor conditioned on the event that the battery is functional during T_m
λ_b	Probability of failure of a SN battery during T_m
$x_i(\boldsymbol{\pi})$	State of SN i at state $\boldsymbol{\pi}$ of $\boldsymbol{S}$
$\boldsymbol{X}_p$	Subset of SNs belonging to $\boldsymbol{S}$ with failed processor
$\boldsymbol{X}_b$	Subset of SNs belonging to $\boldsymbol{S}$ with failed battery
$\boldsymbol{F}_r^k$	Set of all the tolerable SN combinations in the *relay* mode of length k assuming the remainder of SNs are in the *on* mode
$\boldsymbol{F}_r$	Set of all the tolerable SN combinations in the *relay* mode assuming the remainder of SNs are in the *on* mode
$\boldsymbol{F}_o^k$	Set of all the tolerable SN combinations in the *off* mode of length k assuming the remainder of SNs are in the *on* mode
$\boldsymbol{F}_o$	Set of all the tolerable SN combinations in the *off* mode assuming the remainder of SNs are in the *on* mode
R_{min}	Minimum required reliability level in the MCRC-SDP
N_{UB}	Upper bound for the number of connected covers for a given MCRC-SDP instance
$\Phi(\boldsymbol{S}_k)$	Complete-redundancy binary indicative function for a given connected $\boldsymbol{S}_k$
$\boldsymbol{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \dots, \boldsymbol{S}_N\}$, $1 \leq N \leq N_{UB}$	WSN deployment consisting of N non-overlapping connected covers
$R(\boldsymbol{S})$	Reliability of the deployment $\boldsymbol{S}$
$\mathcal{F}(c(N))$	Fitness of chromosome $c(N)$
μ	Number of chromosomes in an MA population
P_{LS}	Ratio of the MA population undergoing the LS procedure
n_{LS}	Number of MA generations at which the entire population undergoes LS
$G(\boldsymbol{V}, \boldsymbol{E})$	Construction graph of the ACO
d_0	Location of the sink node in the RoI of the WSN
$\boldsymbol{S}^a = \{\boldsymbol{S}_1^a, \boldsymbol{S}_2^a, \dots, \boldsymbol{S}_{N_a}^a\}$	WSN deployment consisting of N^a non-overlapping connected covers corresponding to the tour of ant a
p_{ij}^a	Probability of a transitions between deployment points i and j in the tour of ant a
τ_{ij}	pheromone trail value between deployment points i and j
η_j^a	Heuristic value of adding the deployment point j to the connected cover currently being built by ant a
$\boldsymbol{\mathcal{N}}_i^a$	Feasible neighborhood of ant a at its current position in the construction graph at deployment point i
α	Influence parameter of the pheromone trail values
β	Influence parameter of the heuristic information
ρ	Pheromone evaporation rate
$\boldsymbol{D}^-$	Set of deployment points not visited so far by ant a in its current tour
$\boldsymbol{\mathcal{N}}_{ifull}^a$	Set of deployment points within the communication

	range r_c of any deployment point belonging to the current connected cover constructed by ant a
$\boldsymbol{\mathcal{N}}_{ieff}^{a}$	Subset of $\boldsymbol{\mathcal{N}}_{ifull}^{a}$ in which deployment points have a non-zero coverage gain
g_j^a	Coverage gain of adding deployment point j to the current connected cover constructed by ant a
$\boldsymbol{\mathcal{N}}_{sink}^{a}$	Set of deployment points belonging to $\boldsymbol{D}^{-}$ which are within a distance equal to the SN communication range r_c
$\mathcal{C}(\boldsymbol{S}^a)$	Cost of the tour of ant a
$\boldsymbol{S}^{ib}$	Iteration-best solution in the ACO
$\mathcal{C}^{ib}$	Cost of the iteration-best solution in the ACO
τ_{max}	Maximum allowed pheromone value
τ_{min}	Minimum allowed pheromone value
n_{max}	Maximum number of generations/iterations for MA and ACO
n_{conv}	Number of generations before flagging convergence for MA and ACO
t_p	Duration of the RD-TTCP intervals
t_l	Duration of the RD-TTCP listening period
t_h	Frequency of the "heart-beat" messages in the RD-TTCP
t_{ack}	ACK messages time-out duration
$\boldsymbol{S}_{on}$	Currently active connected cover in the RD-TTCP

Contents

Chapter 1

Introduction

1.1. Introduction

Over the past decade, Wireless Sensor Networks (WSNs) have become a rich research field, introducing a wide variety of exciting new applications. A WSN is composed of a number of tiny low-power sensor devices that are capable of sensing various physical phenomena (e.g. sound, light, temperature, motion, seismic action, etc.) in almost all types of environments (industrial, domestic, military, etc.). These devices, simply referred to as Sensor Nodes (SNs), process the crude sensory data and wirelessly communicate it to one or more data collection nodes, referred to as sinks, through single or multi-hop transmissions. The sink(s) are in turn connected to another wired or wireless network for the purposes of querying and processing the collected data [1]. WSNs are also considered one of the key technologies for realizing the Internet of Things (IoT) concept, playing the pivotal role of detecting events and measuring physical and environmental quantities of interest [2], [3]. It is currently estimated that the WSN market will grow to $1.8 billion by 2024 [4]. Current applications for WSNs include but are not limited to industrial near real-time monitoring and automation [5], [6], traffic surveillance and control [7], [8], continuous health monitoring [9] , [10], target tracking in military operations [11] and environmental monitoring [12], [13]. Nevertheless, the large potential of WSNs in vital applications is associated with their highly complex design process. The reason behind the design complexity is that WSNs are inherently different from existing wired and wireless networks due to the severe energy, processing and communication constraints of their constituent SNs.

One of the most important design aspects in WSNs is the deployment of SNs, which is also covered under different expressions in the literature: SNs' positioning, placement, topology construction and deployment. Throughout this dissertation, the expression *deployment* will be used. The importance of SN deployment lies in the fact that it affects almost all the performance metrics of a WSN, such as the connectivity between SNs, the network's effective coverage and the network's effective lifetime. Consequently, a considerable body of research in the field of WSNs has been dedicated to addressing deployment related issues.

In general, SN deployment methods fall under two main categories, namely, planned deployment and random deployment. In random deployment, SNs are usually scattered (e.g. by aircraft), resulting in a randomized distribution of sensors, although their density can be controlled to an extent [14]. Random deployment can sometimes be the only feasible option in some applications where the Region of Interest (RoI) is inaccessible such as disaster areas and active war zones. Logically speaking, random deployments result in sub-optimal performance of the WSN. On the other hand, planned deployment is defined as selectively deciding the locations of the SNs to optimize one or more design objectives of the WSN, under the constraints of a specific application. Hence, planned deployment is often formulated as an optimization problem, which we will refer to here as the SN Deployment Problem (SDP). Design objectives of the SDP commonly required are minimizing the deployment cost, maximizing coverage, minimizing energy consumption, and minimizing

routing costs. Planned deployment is suited for a wide variety of WSNs applications, provided that the RoI is accessible. Examples include border surveillance and intrusion detection in facilities[15], continuous human health monitoring [16], structural health monitoring [17] and industrial real-time monitoring [18]. A great number of studies have proposed methods and algorithms for solving the planned sensor deployment problem. These proposed deployment algorithms in the literature are heavily influenced by the requirements of specific applications and the characteristics of the used SNs, such as their wireless communication ranges and their sensing profiles.

The examples of WSNs applications requiring planned deployment cited above are also examples of WSNs applications of a critical nature. For these applications, it is imperative that the WSN functions properly throughout its *mission time*, i.e. the WSN is *reliable* during this time. This is because failure of the WSN to carry out its required tasks can have serious detrimental effects (e.g. failure to detect an intrusion at a military installation). The mission time for a WSN is application-dependent and can either be the intended lifetime of the network or the time interval between regular network maintenance operations. This in turn poses stringent reliability requirements on the WSN that must be addressed in the design and deployment phase of the network. Consequently, it is important to clearly define proper WSN functionality and to identify the different issues that can compromise it. Moreover, a well-defined reliability metric is required to serve as a measure of the fault tolerance of the WSN deployment against these issues.

In general, the reliability of a multi-component system is defined as the "probability that the system will perform satisfactorily during its specified mission time when used under the stated conditions" [19]. The method by which reliability is calculated for a specific system varies according to the type(s) of components the system is composed of, the system's layout or configuration in terms of how these components are connected to each other and the state(s) at which the system is defined to have failed. In this context, a WSN can be viewed as a multi-component system in which the components are the SNs and the sink node(s). Each SN is characterized by a number of parameters which include the reliabilities of its own components or alternatively, their probabilities of failure during the specified mission time of the network. The layout or configuration of the WSN is defined as the way the SNs are deployed in the RoI and the resulting wireless communication graph, assuming any two SNs can communicate wirelessly if the distance between them is less than their communication range (i.e. assuming SNs act as both sensory data sources and relays in the WSN).

In order to define the states at which a WSN fails, proper WSN functionality must be defined, i.e. the conditions required for a WSN to be functional. The functionality of a WSN can be divided into two major elements. The first element is the *sensing* functionality, which is the ability of a WSN to detect all the targets or phenomena that occur inside the boundaries of the RoI during its mission time. Hence, for a WSN to be functional in terms of sensing, it must provide full coverage for the RoI area (in case of area coverage) or all the targeted locations in the RoI (in case of point coverage) during its mission time. The second element of the WSN functionality is the *connectivity* functionality, which is the ability of the WSN to deliver sensed data from its sources (i.e. SNs) to the designated destination (i.e. sink node(s)) during its mission time. Hence, for a WSN to be functional in terms of connectivity, any target or a phenomenon detected by one or more SNs in a WSN has to be recognized at the sink node(s) through multi-hop wireless communication throughout the WSN mission time. Based on this definition of WSN functionality, a WSN is said to have failed if either of its sensing or connectivity functionality elements fails [20]. Therefore, a WSN is said to be *reliable* during a specified mission time if *both* its functionality elements do not fail during that time interval, i.e. if the WSN provides a *connected-cover* of the RoI throughout its mission time.

1.2. Problem Statement and Research Objectives

Deployment of reliable WSNs is a challenging problem. This is due to the fact that SNs are subject to random failures that result from different sources such as hardware failures, harsh environmental conditions and many other reasons [20]. Naturally, such failures can compromise the WSN functionality in terms of sensing and/or connectivity. Hence, SN redundancy (i.e. the presence of redundant SNs in the network) is essential to guarantee the reliable operation of a WSN during its intended mission time. However, for many applications for which SNs are equipped with expensive hardware minimizing the number of deployed SNs (i.e. the network deployment cost) presents a major concern. Therefore, the level of SN redundancy in the WSN must be carefully quantified such that the network meets the reliability requirements imposed by the application while avoiding any unnecessary increase in the network deployment cost.

After carrying out an extensive survey on the existing static WSN planned deployment algorithms (which is presented and discussed in Chapter 2), we realized that WSN reliability, specifically in conjunction with minimizing the WSN deployment cost, has been overlooked as a deployment objective. The majority of the existing deployment algorithms aim at minimizing the WSN deployment cost with the objective of fulfilling application-dependent network coverage and connectivity objectives under a wide range of assumptions. In other words, they aim to find a connected-cover of the targeted RoI that is *cost-optimal* [21]. However, a WSN which consists of a single connected-cover cannot be considered reliable over a given network mission time. This is because, by definition, a cost-optimal connected cover does not contain redundant SNs. Consequently, the failure of one or more of the deployed SNs during the network mission time will compromise the functionality of the network in terms of coverage and/or connectivity. On the other hand, some recent studies [22] propose deployment algorithms that find cost-optimal deployments that are characterized by higher degrees of coverage and/or connectivity. However, this approach do not offer a method to predict the level of SN redundancy (i.e. the degree of coverage and connectivity) required to guarantee a specified minimum level of network reliability over a given network mission time. Therefore, in order to deploy reliable cost-optimal WSNs, it is important to devise a deployment technique that explicitly considers network reliability as a design requirement while ensuring that the deployment cost is minimized.

Addressing the problem stated above is the central focus of this dissertation. Specifically, we aim to develop WSN deployment algorithms that obtain reliable and cost-optimal deployments for WSNs that support critical applications under practical operational assumptions. We define the reliable WSN cost-optimal deployment as one that has minimum deployment cost with a reliability level that meets or exceeds a given minimum level as specified by the targeted application. We use practical operational assumptions that include arbitrary SN coverage profile, arbitrary SN communication to sensing range ratio and a realistic SN operational model.

The research contributions of this dissertation can be summarized in the following points:

1. A comprehensive survey and classification of the existing WSN planned deployment algorithms based on their underlying mathematical approach.
2. A novel reliability metric for WSNs that is based on an accurate SN model when compared to the model used in the existing WSN reliability metrics.
3. A mathematical formulation of the problem of finding a reliable cost-optimal WSN deployment. We coin this problem the Minimum-Cost Reliability-Constrained Sensor Node Deployment Problem (MCRC-SDP) and prove that it is NP-Complete.

4. A Memetic Algorithm (MA) and an Ant Colony Optimization (ACO) algorithm designed for solving the defined MCRC-SDP.
5. A practical realization of a Topology Control Protocol (TPC) suitable for managing the SN redundancy in the reliable cost-optimal WSN deployments obtained by the developed deployment algorithms.

1.3. Organization of the Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, a classification of the WSN planned deployment algorithms is presented. Based on an extensive review of the deployment algorithms which belong to each approach, we present a qualitative comparison among them highlighting the strengths and shortcomings of each approach. We also present an experimental quantitative comparison of four of the existing static WSN deployment algorithms reviewed in the chapter. In Chapter 3, a novel WSN reliability metric is proposed to address some of the limitations of the existing WSN metrics. Experimental results on the proposed metric are presented and discussed in terms of its computational efficiency and accuracy. In Chapter 4, the mathematical formulation for the MCRC-SDP is presented. A MA and an ACO algorithm designed for solving the defined MCRC-SDP are proposed. Extensive experimental results are presented, analyzed and discussed to highlight the strengths and limitations of each of the proposed algorithms. In Chapter 5, we propose a practical TCP that is suitable for managing the SN redundancy in reliable cost-optimal deployments. We present and discuss the experimental results obtained from implementing and simulating the proposed TCP. Finally we conclude the dissertation in Chapter 6, in which we summarize our findings and propose future work directions.

Chapter 2

Wireless Sensor Network Deployment Techniques: A Survey and Classification

2.1. Introduction

There exist a large number of studies in the literature which propose algorithms for solving different variants of the planned SDP that we defined in Chapter 1. In this chapter, we present an extensive literature survey on these studies. We begin by discussing some of the fundamental design factors of WSNs, namely the SN sensing/coverage model, SN mobility and WSN coverage and connectivity. We elaborate on these specific design aspects due to their great influence on the deployment algorithms that are reviewed in this chapter. We present a novel classification of the WSN planned deployment algorithms, based on the mathematical approach used for modeling and solving the deployment problem. The presented classification encompasses the majority of the main studies conducted on the topic. Four distinct mathematical approaches are presented: Genetic Algorithms (GAs), Computational Geometry (CG), Artificial Potential Fields (APFs) and Swarm Intelligence (SI). For each approach, we provide a discussion of its background and basic mathematical foundation. We then review the algorithms which belong to each approach and provide a comparison between them in terms of their objectives, assumptions and performance. Based on our extensive survey, we discuss the strengths and limitations of the four approaches and compare them in terms of the different WSN design factors. We also present an experimental comparison among four of the existing WSN deployment algorithms.

2.2. Fundamental Design Factors of WSNs

2.2.1. Sensing Model

Generally speaking, the sensing model of a specific type of SNs is a mathematical model that describes the probability of target/event detection of the SN. Assuming the target or event occur at a point p_j in the RoI, the probability of detecting that target or event by a SN s_i is denoted by P_{ij}, which is a function of several parameters. The most commonly used parameters are the Euclidean distance between them d_{ij}, the orientation of the SN (e.g. image SN with a given Field of View (FoV), various environmental parameters and SN hardware parameters. There are several sensing models found in the literature. However, they can be broadly classified into binary and probabilistic sensing models.

2.2.1.1. The Binary Sensing Model

This model is also called the Disk Model. The Binary model simply assumes that a SN has a fixed sensing range r_s . If an event occurs at a point p_j at a distance less than or equal r_s from the location of SN s_i , then the event is detected deterministically by s_i . However, if the distance is equal to $r_s + \epsilon$ ($\epsilon > 0$), then the event won't be detected at all. This definition is depicted in (2.1) [23], [24].

$$P_{ij} = \begin{cases} 1 & if\ d_{ij} \le r_s \\ 0 & otherwise \end{cases} \quad (2.1)$$

Although this model is widely used in the literature due to its simplicity, it is unrealistic. It is unlikely that the detection capabilities, or the physical signals of the detected target or event, drops abruptly from maximum to zero. This implies that using the binary sensing model in a deployment scheme may result in under-utilizing the sensing capabilities of SNs and hence deploying more SNs than needed, incurring higher deployment costs.

2.2.1.2. Probabilistic Sensing Model

Probabilistic sensing models aim at capturing the various factors affecting the precision of SN readings [25]. Apart from the nature of the sensed physical phenomenon and the imperfect detection capabilities of the SN itself, these factors also include environmental conditions such as noise and obstacles [26]. Inspired by the probabilistic sensing models proposed in the field of robot navigation [27], the study in [28] proposed the following probabilistic sensing model:

$$P_{ij} = \begin{cases} 1 & if\ r_s - r_e \ge d_{ij} \\ e^{-\alpha a^{\beta}} & if\ r_s + r_e \ge d_{ij} \ge r_s - r_e \\ 0 & if\ r_s + r_e \le d_{ij} \end{cases} \quad (2.2)$$
$$a = d_{ij} - (r_s - r_e),$$

where r_e ($r_s > r_e$) is the measure of the uncertainty in the SN detection and α and β are SN parameters with values between 1 and 0, varying according to the physical characteristics of the SN. The model depicted in (2.2) assumes that there are two concentric circles around a given SN; a circle of confidence with a radius $r_s - r_e$ in which the detection probability of a target is equal to 1, and a wider circle of radius r_s. The probability of detecting a target outside the circle of confidence and inside the wider circle deteriorates exponentially with the Euclidean distance between the SN and the target or event d_{ij}.

A simpler variation of the sensing model described by (2.2), proposed in [29], omits the inner circle of confidence, and simply assumes an exponential decay of the probability of detection with the Euclidean distance, as depicted in the following equation.

$$P_{ij} = \begin{cases} e^{-\beta d_{ij}} & if\ d_{ij} < r_s \\ 0 & otherwise\ , \end{cases} \quad (2.3)$$

Equations (2.1) to (2.3) describe the sensing models that are most commonly adopted in the formulation of the deployment algorithms reviewed in this chapter. Hence, these equations will be referenced repeatedly throughout this chapter.

2.2.2. Sensor Node Mobility

WSNs can be classified according to the type of deployed SNs (static or mobile) into two types: Static WSNs (simply referred to simply as WSN) and Mobile WSNs (MWSNs). The concept of MWSNs has been spurred by the recent advances in distributed computing and robotic technology. MWSNs are defined as WSNs containing SNs which have sensing, processing, communication and locomotive capabilities. The locomotive capabilities can be achieved by mounting static SNs on mobile vehicles or mobile robots [30], [31]. The MWSN

can be homogeneous, i.e. consisting entirely of mobile SNs, or heterogeneous, i.e. contains both mobile and static SNs.

The added mobility can provide the MWSNs with several benefits, but can also complicate the design significantly. The main added benefit is the ability of the network to self-deploy after an initial random deployment. This *self-deployment* (or more accurately re-deployment) capability can significantly improve the effective coverage of the network from the initial limited coverage which is difficult to control, due to factors such as wind, foliage, terrain irregularities, etc. Mobile SNs can also be used to cover detected coverage holes in a RoI in heterogeneous WSNs consisting mostly of static SNs as proposed in [32] and [33]. Another important benefit is the added ability of self-reconfiguration in the network. Re-configuration can be very beneficial in the case when some of the SNs in the network die (due to energy depletion or harsh environmental conditions) which leads to connectivity islands. Re-configuration will enable the network to maintain an acceptable connectivity, in order to maintain multipath routing [34]. However, the introduced mobility poses design problems such as the hefty burden on the limited energy resources of the usually battery-powered SNs and the need for coordination between the mobile SNs.

2.2.3. WSN Coverage and Connectivity

Coverage is one of the most important performance metrics in WSNs. In WSNs, coverage can have different meanings and can be determined using different methods. In general, it revolves around the question "How well do the deployed SNs observe the physical space?" In all contexts, coverage provides a measure of the Quality of Service (QoS) of a specific WSN deployment. As the coverage of a WSN increases, its success in carrying out its specific sensing task(s) increases as well.

Coverage in a WSN is intertwined with the connectivity in the network. The term *k-connectivity* ($k \geq 1$) means that there exist at least k disjoint paths between any pair of SNs in a WSN [23]. Connectivity is of immense importance because it guarantees that sensory data acquired by any SN in a WSN can be routed to the sink node(s).

Coverage in WSNs can be classified into three types: area (blanket) coverage, point coverage and barrier coverage [35]. In area coverage, an entire two dimensional (2-D) RoI is considered, such that each point in the RoI is observed by at least one SN. In point coverage, the objective is to only guarantee that a set of finite points in the RoI are observed by at least one SN. Barrier coverage usually deals with the detection of movement across a barrier of SNs. The most studied coverage problem in WSNs literature is the area coverage problem. This is indeed justified, since the majority of WSNs applications involve full monitoring.

The two main problems pertaining to area coverage are achieving satisfactory coverage and evaluating the coverage for a certain deployment. Satisfactory coverage here means ensuring that an event occurring at any point in the RoI can be detected with a probability that is equal to or exceeds a predefined threshold γ dictated by the application, where $\gamma \in [0,1]$. Another method of describing satisfactory coverage is to specify the minimum number of SNs k observing each point in the region of interest, which is denoted *k-coverage.*

The methods used for achieving the required level of area coverage depend primarily on the deployment method (planned or random), the type of the application (which determines the value of γ), and the adopted sensing model for the deployed SNs. For example, consider the ideal case of observing an obstacle-free 2-D RoI, assuming a binary sensing model, where planned deployment is feasible. In such a hypothetical case, it is proposed that SNs be deployed in regular deployment patterns, such as triangular lattice patterns, square grid patters and hexagonal patterns [36], [37]. The objective of these regular deployment patterns is to ensure that area coverage is achieved with no coverage holes while also ensuring k-

connectivity. The dimensions of the lattice (distances between SNs) depend on both the sensing radius $r_s$and communication range r_c of the SNs. It has been proven that in the case where $r_c \geq 2r_s$, k-coverage implies k-connectivity, assuming a binary sensing model [38]. Moreover, the authors in [39] propose strip-based deployment patterns to achieve 1- or 2-connectivity for different values of the communication to sensing range ratio r_c/r_s. However, such regular planned deployments require precise manual deployment of SNs, which can be impractical or even impossible in most applications; e.g. when the RoI is characterized by static and/or dynamic obstacles such as in indoor environments, mountainous outdoor environments and forests. This has motivated the emergence of the different approaches for planned deployment surveyed in this study, in which achieving satisfactory area coverage is always a primary objective.

On the other hand, evaluating the effective area coverage of an existing WSN deployment has also a paramount importance. For example, consider WSNs applications were only a random deployment is feasible, such as surveillance applications in war zones or catastrophe areas. In these applications, if the number of SNs deployed randomly is high, i.e. dense deployment, the need arises for protocols to control the activation and deactivation of the deployed SNs, in order to minimize the redundancy in coverage (overlap of sensing regions of neighboring SNs) in some areas of the RoI. This results in increasing both the network's lifetime and fault tolerance. These protocols are called *coverage protocols* in the WSN literature. Examples include Optimal Geographical Density Control (OGDC) protocol [36], Coverage Configuration Protocol (CCP) [38], and Probabilistic Coverage Protocol (PCP) [23]. In these coverage protocols, probabilistic sensing models are assumed, as depicted in (2.2) and (2.3). In order to compute the effective area coverage achieved by a set of n active nodes $\boldsymbol{S}$, a sampling method is used where the RoI is represented in the form of a 2-D grid containing a finite number of m grid points. The collective probability of detecting a target or an event at a grid point p_j , $j \in \{1, 2, \dots, m\}$, from all n active SNs in $\boldsymbol{S}$ is given by:

$$P(p_j) = 1 - \prod_{i=1}^{n}(1 - P_{ij}) \,, \tag{2.4}$$

where P_{ij} is calculated using (2.2) or (2.3). It is then assumed that the whole RoI is adequately covered if the probability of detection at each grid point exceeds the predefined threshold, γ. This sampling approach for evaluating the area coverage of a WSN deployment is also utilized in grid-based deployment algorithms [29], [40], [41], where SNs can only be placed in a subset of m predefined grid points that represent the RoI. The major drawback of these grid-based deployment algorithms is that their accuracy and computational complexity are dependent on the number of grid points considered in the RoI. Other sampling approaches which utilize famous computational geometry constructs such as Voronoi diagrams have also been utilized in non-grid based deployment algorithms, with less complexity, as discussed later in this chapter.

2.3. Mathematical Approaches Used in WSN Deployment Algorithms

On reviewing the plethora of planned deployment and re-deployment algorithms proposed for WSNs in papers published in the past decade, we identified four mathematical approaches or *tools* commonly used for building such algorithms. In this section, we aim to provide the reader with the necessary background and foundations of these mathematical approaches.

2.3.1. Genetic Algorithms

Genetic Algorithms (GAs) are search and optimization algorithms based on the mechanics of natural selection and genetics. GAs became popular through the work of John Holland in the early 1970s [42], and have since been used for solving optimization problems in various fields such as computer networking, industrial engineering and machine learning. The paradigm of GAs is copying the natural selection as described in Darwin's Theory stating that "species whose individuals are best adapted survive; others go extinct". A GA can be especially effective in combinatorial and multi-objective optimization problems, in which deterministic optimization methods are not applicable [43]. In general, a GA has three basic components [44]:

1. A genetic representation of the candidate solutions of the problem. This is called encoding and it is dependent on the problem's variables and constraints. The encoding of candidate solutions is chosen in such a way that they can be decoded into a unique variables' vector which belong to the search space and verify the constraints. There are several methods for encoding in GAs, such as binary encoding, integer encoding, and real number encoding. The choice of encoding method is highly dependent on the nature of the optimization problem itself. The candidate solutions in the problem's search space are said to be in the phenotype space, while their genetic representation through encoding is in the genotype space.
2. A fitness function for evaluating candidate solutions.
3. Stochastic genetic operators that alter the composition of the offspring during the reproductive phase of the GA.

Five steps are carried out in a single iteration of a typical GA. The first step is creating an initial *population* of individuals or *chromosomes*. Each chromosome represents a unique encoded candidate solution of the problem. The initial population usually covers the search space of the problem uniformly.

After creating the initial population, step two is carried out; using a fitness function to evaluate the individuals in the population. The fitness function is essentially a cost function, which is a mathematical expression of what we want to optimize. GAs use fitness evaluation for the elimination of the weakest individuals from the population and to find out the fittest individuals. Therefore, if a chromosome brings the fitness evaluation to a value closer to the optimal point than the others, that chromosome is said to be the fittest.

The third step is selecting chromosomes from the population to undergo the reproductive phase of the GA, this is called *parent selection*. Parent selection is usually dependent on the calculated fitness and is often stochastic in nature. The two most commonly used parent selection techniques are the Roulette Wheel and the Tournament techniques.

The fourth step is then to apply the two genetic operators, *crossover* and *mutation*, to the selected parent chromosomes to produce an *offspring* or *children* population Crossover is the primary genetic operator, and is achieved by randomly pairing every two individuals (*parents*) in the population together to produce offspring that contain portions of both their codes. Fig. 2.1 illustrates the process of crossover for binary encoding. Mutation, on the other hand, is a background operator that creates a new individual by altering a randomly chosen part of a selected parent.

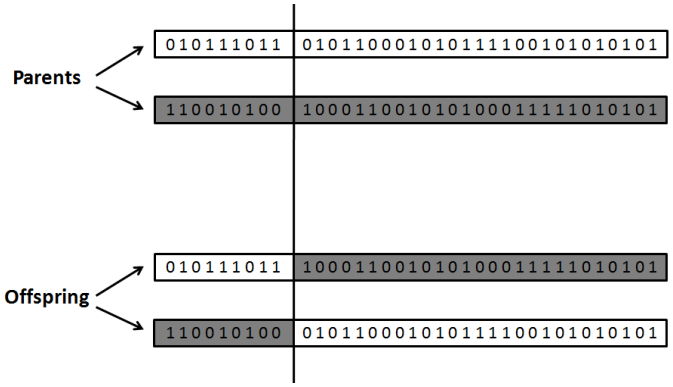


Fig. 2.1 Crossover in case of binary encoding in a GA

Table 2.1 Pseudo code of a general GA

Step	Genetic Algorithm
1.	Set $t \leftarrow 0$
2.	**Initialize** $P(t)$
3.	**Evaluate** $P(t)$
4.	**While** (termination condition not met)
5.	**Recombine** $P(t)$ to yield $C(t)$
6.	**Evaluate** $C(t)$
7.	**Select** $P(t+1)$ from $P(t)$ and $C(t)$
8.	$t \leftarrow t+1$
9.	**End While**

Both operators are responsible for directing an offspring population towards exploring new parts of the problem's search space. The offspring population is also evaluated using the same fitness function.

The fifth step in a GA is the *selection* step, which is choosing individuals from both the parent and offspring populations to form a new population. Selection is the driving force of the GA, since it direct the search to promising regions of the search space. There are several methods for selection, both stochastic and deterministic, such as age-based selection, fitness-based selection and elitism. Steps two to five are repeated to produce several iterations, or *generations*, and the algorithm gradually converges to the fittest individual, which hopefully represents an optimal solution to the problem, although that outcome isn't guaranteed. The algorithm can either terminates after producing a maximum number of generations or after finding an individual with a fitness corresponding to a satisfactory solution to the problem. The general structure of GAs is expressed in pseudo code in Table 2.1.

One of the most important advantages of GAs as an optimization tool is its ability to deal with combinatorial and multi-objective optimization problems. This advantageous property of GAs prompted their use in formulating multi-objective deployment algorithms for WSN. In multi-objective GAs, or MOGAs, one of the approaches to measure fitness is to use a sum of weighted normalized cost functions of each objective independently [43] as expressed by the following equation:

$$Cost = \sum_{i=1}^{N} w_i f_i \,, \qquad (2.5)$$

$$0 \leq f_i \leq 1 \,,$$

where f_i is the i^{th} normalized cost function ($1 \leq N \leq i$) and w_i is the weighting factor where $\sum_{i=1}^{N} w_i = 1$. Another approach for fitness evaluation in MOGAs is the Rank-based Fitness assignment [45]. It depends on the concept of *Pareto Dominance* which can be explained as follows: Given a set of objectives in a MOGA, an individual is said to *Pareto dominate* another if the first is not inferior to the second in any of the objectives, and there is at least one objective where it is better. Consequently, the optimal solution of the multi-objective optimization problem isn't represented by a single individual, but by a set of *non-dominated Pareto optimal* individuals. In Rank-based fitness assignment, individuals are given *ranks* that are directly proportional to the number of individuals dominating them.

2.3.2. Computational Geometry

The field of Computational Geometry (CG) emerged in the 1970s. It dealt with various kinds of challenging computational problems of geometric nature. Examples of such problems include motion planning in the field of robotics, map overlay in geographic information systems and polygon triangulation which is used to solve the famous Art Gallery Problem in surveillance applications. These versatile geometric problems motivated researchers to come up with carefully designed, efficient and fast geometry-based algorithms to solve them. A formal definition for CG is given as "The systematic study of algorithms and data structures for geometric objects, with a focus on exact algorithms that are asymptotically fast"[46]. Nowadays, there is a rich collection of geometric algorithms that are efficient and easy to understand and implement for various application areas.

In the field of WSNs, several studies were based on two of the famous CG structures, the Voronoi Diagram (VD) and Delaunay Triangulation (DT). They aimed at constructing efficient deployment algorithms for both static and mobile WSNs. In the following we shed some light on each of these two geometric structures.

2.3.2.1. Voronoi Diagram

The VD is a versatile geometric structure with applications in physics, astronomy, robotics and networking [47]. It is closely linked to DT which will be explained in the next subsection.

To formally define a VD [46], consider a set of n distinct points in a set P in a 2-D plane, such that $P = \{p_1, p_2, \dots, p_n\}$. These points are called *sites* in the terminology of CG. The VD of the set P is defined as the subdivision of the 2-D plane into n cells, where each cell corresponds to one site in P, such that the Euclidean distance between any site p_i and any point q lying inside the cell corresponding to it is less than that between q and another site p_j, for $i \neq j$. This is expressed in (2.6) as:

$$dist(q, p_i) < dist(q, p_j) \quad \forall\, p_i, \qquad p_j \in P, \qquad i \neq j \tag{2.6}$$

The VD of P is denoted $\mathrm{Vor}(P)$, while the Voronoi cell (also called Voronoi *polygon*) corresponding to site p_i is denoted $v(p_i)$. To construct $v(p_i)$, we draw the cell's edges as the vertical bisectors of the lines connecting p_i to its neighboring sites. This is illustrated in Fig. 2.2, where the Voronoi polygon of point O, $v(O)$ is constructed by drawing the vertical bisector of the lines passing through point O and points *A,B,C,D* and *E* (neighboring sites).

One of the ways VDs were utilized in the field of WSNs is using them as a means of computing the effective coverage of a 2-D RoI. It is a sampling method, like using grids, to check a finite number of points inside and on the boundary of the RoI for coverage, to evaluate the area coverage of the entire RoI [48] - [52]. Using a VD for this purpose, the sites

correspond to deployed SNs and the vertices of the Voronoi cell of each SN are the sampling points checked for coverage, along with a number of points on the boundaries of the RoI. If the coverage of the WSN at the sampling points exceeds the minimum required coverage, then it is guaranteed that the entire RoI is covered, following the VD property expressed in (2.6). Examples of the use of VD in deployment algorithms for WSNs are reviewed at length in Section 2.4.2.

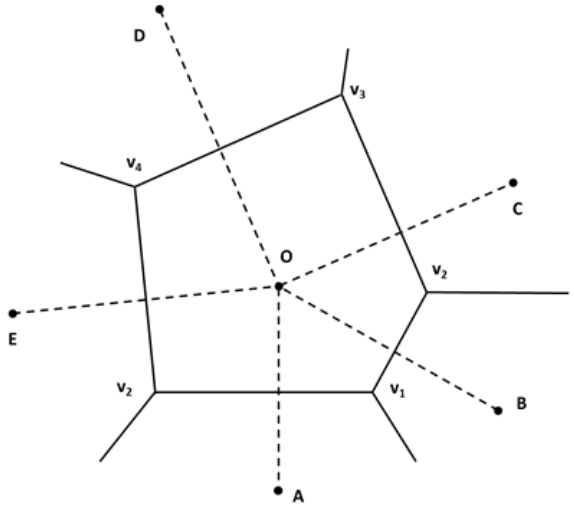


Fig. 2.2 Voronoi cell of a site O with neighbouring sites A,B,C,D and E

2.3.2.2. Delaunay Triangulation

In general, *triangulations* of planar point sets are widely used in approximating the earth's terrain in topographic maps, using measured heights at a finite set of sample points. Triangulation of a set P is defined as "the planar subdivision whose bounded faces are triangles and whose vertices are the points of the set P" [46]. The DT is a special kind of triangulation of planar point sets. The DT of a set P is always an angle-optimal triangulation of P, which essentially means that it would give the most realistic approximation of a certain terrain, compared to other possible triangulations.

As mentioned earlier, DT is closely related to VD. To construct the DT of a set P, denoted $DT(P)$, consider $Vor(P)$, as shown in Fig. 2.3. $DT(P)$ is constructed by connecting every two sites in P if their corresponding Voronoi cells share an edge, i.e. neighboring sites. $DT(P)$ has a very interesting property; the circumcircle of any triangle in $DT(P)$ does not contain a point of P in its interior. This property is called the *empty circle* property, and as we will discuss later in Section 2.4.2, it can help estimate the point of the weakest coverage in a deployed WSN, and hence provide very useful guidance in the case of deploying new SNs (or waking them up from a sleep state) to improve the effective coverage.

In addition to computing effective area coverage and discovering coverage holes in WSNs, VD and DT were also used in determining the Maximal Breach Path (MBP) and Maximal Support Path (MSP) in a certain WSN deployment [53],[54]. MBP corresponds to the worst-case coverage. It is defined as the path between two arbitrary points that passes through a WSN with a bounded RoI, such that the distance between each point on the path and the nearest SN is maximized. On the other hand, MSP corresponds to the best- case coverage, where the distance between each point on it and the nearest SN is minimized. Both MBP and MSP are used in determining barrier coverage, which was defined earlier in Section 2.2.5.

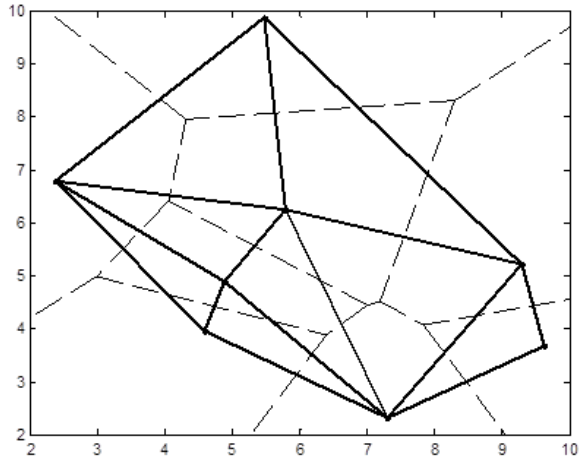


Fig. 2.3 Delaunay triangulation of a set of planar points

2.3.3. Artificial Potential Field

Artificial Potential Field (APF) techniques were first introduced in the field of Robotics in [55]. The study presented an original real-time obstacle avoidance approach for mobile robots based on an "artificial potential concept". The idea behind the approach can be described as follows. A mobile robot is assumed to be moving in a field of artificial virtual, forces. The position to be reached, i.e. the goal, can be represented by an attractive pole, which exerts virtual attractive forces on the mobile robot. The obstacles are represented by repulsive surfaces that in turn exert virtual repulsive forces on the mobile robot. This approach is mathematically interpreted in the following equation:

$$U_{art}(x,y) = U_g(x,y) + U_o(x,y)\,, \tag{2.7}$$

where $U_{art}(x,y)$ is the artificial potential energy that varies with the location of the mobile robot in the field, $U_g(x,y)$ is the artificial attractive potential energy attributed to the goal and $U_o(x,y)$ is the artificial repulsive potential energy due to the obstacles. The virtual force vector $\boldsymbol{F}$ applied on a mobile robot at a certain location (x,y) in the APF is computed by obtaining the gradient of $U_{art}(x,y)$ in (2.7) as follows:

$$F = -\nabla U_{art}(x,y)\,, \tag{2.8}$$
$$F = F_g + F_o\,, \tag{2.9}$$
$$F_g = -\nabla[U_g(x,y)]\,, \tag{2.10}$$
$$F_o = -\nabla[U_o(x,y)]\,, \tag{2.11}$$

where $\boldsymbol{F_g}$ is the virtual attractive force enabling the mobile robot to reach the goal position, while $\boldsymbol{F_o}$ represents a virtual repulsive force that steers the mobile robot away from the obstacles. The minus sign in (2.8) - (2.11) means that the virtual forces are in the direction of the steepest decrease of the artificial potential fields at any given point (x,y).

This real-time approach aims at making obstacle-avoidance in robotics a component of the low level control that provides a robot with a path to accomplish its assigned goal free from any risk of collision, even in cluttered dynamic environments. The authors in [56] used a variant of the APF method to produce appropriate velocity and steering commands for a

mobile robot as part of a new concept in mobile robot navigation and object tracking called "motor schema". APF techniques have since been applied to the problems of formation control and obstacle avoidance in multi-robot systems [57] - [59]. These problems are of similar nature to the deployment problem in MWSNs; the movement of robots, based on local sensing and computation, collectively maintains a design objective, which is the desired formation shape, while avoiding colliding with obstacles and each other.

In [60], authors mapped the concept of APF to the domain of WSNs. They used the concept to devise a deployment approach for MWSNs. Their deployment algorithm, along with several algorithms based on the APF approach, is discussed in detail in Section 2.4.3.

2.3.4. Swarm Intelligence (SI)

Swarm Intelligence is a branch of Artificial Intelligence (AI) that focuses on the collective behavior and properties of complex, self-organized, decentralized systems with a social structure, such as bird flocks, ant colonies and fish schools. These systems consist of simple interacting agents organized in small societies, called swarms, which exhibit traits of intelligence, such as the ability to react to environmental threats and decision making capacities [61], [62].

Swarm Intelligence was utilized in the global optimization framework in the form of a set of algorithms introduced in [63] for controlling robotic swarms in 1989. Several years later, three main swarm intelligence optimization algorithms were developed, namely, Ant Colony Optimization (ACO), Stochastic Diffusion (SD) and Particle Swarm Optimization (PSO). In this study, we will only focus on PSO and ACO due to their emerging use in the development of deployment algorithms for WSNs.

2.3.4.1. Particle Swarm Optimization

In 1995, Eberhart and Kennedy [64] developed PSO as a stochastic global optimization algorithm based on social simulation models. The core idea of the PSO algorithm is to use a population (*swarm*) of search points (*particles*) that move stochastically in the boundaries of the optimization problem's search space. The nomenclature was inspired from similar models in social sciences and particle physics. The best position (i.e. the best solution) ever reached by each individual in the population, which is called *experience*, is retained in memory. This experience is then communicated to part or all of the swarm, directing its movement towards the search space regions where it is more likely to find the optimal solution. The convergence of the algorithms depends greatly on the chosen communication scheme.

The mathematical framework of PSO [61] is as follows. Let $A \subset \mathbf{R}^n$ ($\mathbf{R}^n$ is the n dimensional space) be the search space and $f: A \to Y \subseteq \mathbf{R}$ be the objective function of the optimization problem, where Y is the corresponding value of f to any point in A. Assume that there are no further constraints in the problem and that there are no other conditions on either A or f. The swarm $\boldsymbol{S}$ is defined as a set of N particles, representing candidate solutions:

$$S = \{p_1, p_2, \dots, p_N\}, \tag{2.12}$$

$$p_i = (p_{i1}, p_{i2}, \dots, p_{in}) \in A, \quad i = 1,2, \dots N, \tag{2.13}$$

where N is a user-defined parameter in the algorithm. The particles are assumed to move within A iteratively in order to explore its promising regions. This is achieved by defining the velocity of each particle, which is used to adjust the particle's position in each iteration t of the algorithm, as follows:

$$v_i = (v_{i1}, v_{i2}, \dots, v_{in}), \quad i = 1,2, \dots N, \tag{2.14}$$

The particle velocity v_i also changes iteratively in the algorithm. The current position and velocity of the i-th particle are denoted $x_i(t)$ and $v_i(t)$ respectively. The algorithm also maintains a memory set, where each particle stores the best position (i.e. best solution) it has ever reached during its search in A. The PSO memory set is given by:

$$M = \{m_1, m_2, \dots, m_n\}, \tag{2.15}$$

$$m_i = (m_{i1}, m_{i2}, \dots, m_{in}) \in A, \quad i = 1,2, \dots N, \tag{2.16}$$

Determining m_i at any iteration $m_i(t)$ depends on the objective function f. The best visited position in A by any particle in the swarm at a given iteration, i.e. the best position in M, is denoted $m_g(t)$. This term represents the social behavior in PSO since particles are assumed to communicate their experiences with each other. The early version of PSO by Eberhart and Kennedy [64] is defined by the following equations:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 R_1 \left(m_{ij}(t) - p_{ij}(t)\right) + c_2 R_2 (m_{gj}(t) - p_{ij}(t)) \tag{2.17}$$

$$p_{ij}(t+1) = p_{ij}(t) + v_{ij}(t+1), \tag{2.18}$$

for $i = 1,2, \dots N$ and $j = 1,2, \dots n$; R_1 and R_2 are random variables uniformly distributed between 0 and 1; c_1 and c_2 are weighting factors; also called the cognitive and social parameter respectively. The steps of the PSO are provided in pseudo code in Table 2.2. It should be noted that PSO has undergone many refinements to its earliest version, as represented by (2.17) and (2.18), to enhance its performance in more complicated optimization problems [65] - [67] . However, the main steps in its operation remain unchanged.

2.3.4.2. Ant Colony Optimization

On the other hand, ACO is based on the fact that ants have the natural capability of finding the shortest path to food using a natural chemical called "pheromone", which the ants lay on the paths they take as they move. ACO algorithms were initially designed by Dorigo, Colorni and Maniezzo to find optimal solutions for the famous traveling salesman problem [68], especially for large instances of the problem. Today, ACO can be used for any optimization problem that can be reduced to finding optimal paths through graphs.

The mathematical framework of ACO is as follows [62], [68]. Let $G(\boldsymbol{V}, \boldsymbol{E})$ represent a directed graph, where $\boldsymbol{V}$ is the set of vertices of the graph and $\boldsymbol{E}$ is the set of edges connecting these vertices. The vertices in $\boldsymbol{V}$ represent different parts/building blocks of a solution to the optimization problem at hand. We assume that the ant colony is a set of k ants $A = \{a_1, a_2, \dots, a_k\}$. A complete solution to the optimization problem is constructed by random walks, called *tours*, of the ants, i.e. a complete solution is a subset of $\boldsymbol{V}$. The ants' tours through $G(\boldsymbol{V}, \boldsymbol{E})$ is influenced by the positive pheromone values associated with every edge $(u, v) \in \boldsymbol{E}$, denoted $\tau_{u,v}$. The tours are also influenced by the heuristic information assigned to every edge, denoted by $\eta_{u,v}$, which is calculated using a cost function that depends on the objective function and constraints of the optimization problem. Assuming that an ant is at vertex $\in \boldsymbol{V}$, a set of allowed successor vertices denoted by $\boldsymbol{N}(u)$ is computed based on the

Table 2.2 Pseudo code of a PSO algorithm

Step	Particle Swarm Optimization
1.	Set $t \leftarrow 0$
2.	**Initialize** S and **Set** $M \equiv S$
3.	**Evaluate** S and M; **Define** index g for best position
4.	**While** (termination condition not met)
5.	**Update** S using (2.17) and (2.18)
6.	**Evaluate** S
7.	**Update** M; **Redefine** index g
8.	$t \leftarrow t + 1$
9.	**End While**
10.	**Print** best position found

constraints of the problem. The probability that the ant chooses vertex $v \in \boldsymbol{N}(u)$ to visit next is given by the following function:

$$p_v = \frac{[\tau_{u,v}]^{\alpha}[\eta_{u,v}]^{\beta}}{\sum_{w \in N(u)}[\tau_{u,w}]^{\alpha}[\eta_{u,w}]^{\beta}}\ , \tag{2.19}$$

where the parameters $\alpha, \beta \geq 0$ are user-defined and determine the relative influence/importance of the pheromone values and the cost function in directing the tours. A complete iteration of the algorithm is concluded when all the ants in A complete their tours, i.e. construct complete candidate solutions of the problem. Before a new iteration is started, the pheromone levels associated with all the edges in $\boldsymbol{E}$ are updated based on the following pheromone-update rule:

$$\tau_{u,v}{}' = \tau_{u,v}(1-\rho) + \sum_k \Delta_i\ , \tag{2.20}$$

where $\tau_{u,v}{}'$ is the updated pheromone level on the edge $e = (u, v) \in \boldsymbol{E}$, the parameter ρ is called the pheromone evaporation factor, where $0 < \rho \leq 1$ and Δ_i is the value of pheromone deposited by ant $a_i \in A$ on $e = (u, v)$ during its tour. The pheromone evaporation factor introduced in (2.20) is responsible for continuously decreasing the pheromone levels on the edges in order to help the algorithm escape local optima. On the other hand, the second term in (2.20) is responsible for increasing the pheromone levels on the edges based on the quality of the constructed solutions in the last iteration, measured by the objective function of the optimization problem, which is sometimes referred to as the *fitness function*. Consequently, the pheromones levels of the edges included in the solutions with higher fitness are higher than those of less fit solutions after the update, giving these edges a higher chance of being included in tours in the next iterations. The algorithm terminates either after a given number of iterations or when a solution with the desired fitness or higher is obtained. Steps of the ACO are illustrated by the pseudo code in Table 2.3.

Table 2.3 Pseudo code of an ACO algorithm

Step	Ant Colony Optimization
1.	Set $t \leftarrow 0$
2.	**Initialize** ants' tours
3.	**While** (termination condition not met)
4.	**Do Until** (a_i completes a tour $\forall\, i = 1, \dots, k$)
5.	Update tours using (2.19)
6.	**End Do**
7.	**Update** pheromone levels using (2.20)
8.	$t \leftarrow t + 1$
9.	**End While**
10.	**Print** best position found

2.4. Wireless Sensor Networks Deployment Algorithms

According to the classification of the different mathematical approaches used in WSNs deployment algorithms outlined in Section 2.3, we now review these algorithms, including their assumptions, objectives and performance.

2.4.1. Genetic Algorithms

Several deployment methods based on GAs were presented in the literature. These algorithms typically aim to optimize the layout of a WSN, with usually more than one deployment objective.

The study in [69] presents a Multi Objective GA (MOGA) for optimal deployment of n static SNs in a 2-D flat RoI, with two competing deployment objectives: maximizing the area coverage and maximizing lifetime. The binary sensing model, as expressed in (2.1), is assumed and all SNs are assumed to have the same communication and sensing ranges, r_c and r_s respectively. For the predetermined number of n deployed SNs, candidate solutions (i.e. deployments) of the problem are represented by a deployment vector $\boldsymbol{DV}$, which contains the coordinates of each SN:

$$\boldsymbol{DV} = [x_1, y_1, \dots, x_n, y_n]\,, \tag{2.21}$$

Rank-based fitness assignment, defined earlier in Section 2.3.1, is used in the algorithm. Each deployment vector (i.e. candidate solution) was ranked according to its area coverage and lifetime. The authors used real number encoding of $\boldsymbol{DV}$, random single-point crossover and a mutation probability of 0.1. The algorithm was tested under different ratios of sensing to communication range (r_s/r_c). Results showed samples of obtained non-dominated pareto-optimal deployments after reaching a maximum number of generations, demonstrating the tradeoff between coverage and lifetime (as coverage increases, lifetime decreases). Results suggested that for the pareto-optimal deployments with maximum coverage, the ratio r_s/r_c affects the shape of the pareto-optimal deployment regarding the extent of overlap in SNs' coverage.

The authors extend their work in [70], where the same MOGA is used, but applied to three specific surveillance scenarios. Each scenario had its own set of competing design

objectives depending on the nature of the surveillance required. The first scenario has three objectives: maximizing coverage, minimizing the number of SNs deployed and maximizing the distance between the deployed SNs and the hostile building under surveillance for maximum survivability of the network. The second and third scenarios require maximizing coverage while minimizing the number of SNs deployed. The difference between the second and third scenarios lies in the type of coverage; barrier coverage is maximized in the second case while area coverage is maximized in the third one. Results show samples of the pareto optimal set of non-dominated deployments for each scenario, obtained after 300 generations of the proposed MOGA. It is shown how these results provide the network designer with trade-off information between the competing objectives.

Results presented in [69] and [70] suggest that the proposed algorithm is flexible in the sense that it can be applied to other scenarios with different sets of design objectives. However, there are two drawbacks that should be pointed out. The first drawback is the use of the binary sensing model in the algorithm. Although it simplifies the computation of coverage, it can lead to misleading results. The second drawback is that the modeling assumes RoIs with a flat terrain, i.e. with no obstacles, which is an unrealistic assumption.

The study in [24] addresses the problem of covering a finite set of target locations, called *target points*, with the minimum number of SNs. We refer to this deployment problem as the *Minimum Cost Coverage* SDP (MCC-SDP). A finite set of possible deployment locations, or *deployment points*, is also assumed to be a given of the problem. The authors propose solving the problem using a GA with two deployment objectives; minimizing the number of deployed SNs (i.e. deployment cost) and ensuring coverage of all target points. To define the problem, a one-zero coverage matrix (a_{ij}) of size ($m \times n$) is used. Each row i in the coverage matrix represents a target point, and each column j represents a combination of three deployment parameters (SN type, deployment point, SN orientation) called *deployment-tuple* and denoted d_j. Two types of SNs are considered, acoustic SNs and image SNs. The acoustic SNs follow the binary sensing model expressed in (2.1), while for image SNs, a FoV metric is used. The deployment problem is then formulated as the following optimization problem:

$$Minimize \ \sum_{j=1}^{n} c_j x_j \ , \tag{2.22}$$

$$subject\ to \ \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i = 1, \dots, m \ , \tag{2.23}$$

$$x_j \ \in \{0,1\}, \quad j = 1, \dots \dots, n \ , \tag{2.24}$$

where c_j is the cost of deploying the deployment-tuple j and x_j is a binary variable that determines whether the deployment-tuple is actually deployed ($x_j = 1$) or not ($x_j = 0$). The constraint in (2.23) guarantees that each row (target point) is covered by at least one column (deployment-tuple).

A candidate solution is simply a subset of all possible deployment-tuples. Candidate solutions are binary encoded and a simplified version of GA, called the Microbial GA [71], was used. A fitness function $f(x)$ is used to evaluate candidate solutions based on the overall cost of deployment as expressed in (2.22), plus a weighted penalty for not covering target points. This fitness function is depicted in following equation:

$$f(x) = \sum_{j=1}^{n} c_j x_j + w(m - coverage) \ , \tag{2.25}$$

where w represents the weight of the penalty, m represents the total number of target points considered in the problem and $coverage$ represents the number of target points covered by a specific deployment-tuple d_j.The authors evaluate their algorithm using a simplified

deployment problem which is constituted of six target points and five deployment points, forming a security SN fence. The problem is simple enough for deriving the optimal solutions deterministically by generating all the possible combinations, and it matches the outputs of the proposed algorithm.

The algorithm proposed in [24] is extended in [72] to include probabilistic coverage determination methods, i.e. the terms of the coverage matrix a_{ij} can take any value between 0 and 1, depending on the probability of coverage of target point i by deployment-tuple d_j. The sensor detection probability used in [72] follows the probabilistic sensing model expressed in (2.2). The same GA was used to find optimal solutions for the deployment problem expressed in (2.22) – (2.24), plus a new constraint given by:

$$p_d(a_i) \geq \alpha \,, \tag{2.26}$$

where $p_d(a_i)$ is the probability of detection of a target point i and α is the required coverage threshold. The proposed deployment GA is tested on a very similar scenario to that in [24] and the optimal solution (chromosomes) were derived using the algorithm matched the ones obtained deterministically by evaluating all combinations. However, the authors in [24] and [72] present a limited case study and didn't apply the algorithm on a more computationally extensive problem to evaluate its computational efficiency. Also, their choice of binary encoding is not practical for more complex scenarios of the deployment problem. This is because their chromosome size is equal to the number of all possible combinations of the deployment-tuples, which means that both the chromosome and population sizes can get very large for practical deployment scenarios involving hundreds of SNs. This in turn means that the algorithm would become very slow and computationally inefficient.

In [73], the authors study the same deployment problem as the study in [24] under the same assumptions. The authors propose an Integer-Encoded Fixed Length Chromosome GA (iFLGA) to solve the MCC-SDP, where all chromosomes have the same number of integer-valued genes throughout the operation of the algorithm. Each chromosome is set to be a permutation of the integers in the interval $[1, n]$, such that every gene represents a deployment point in the RoI. Chromosome decoding to candidate solutions of the problem follows a first-fit approach. That is, starting at the first gene, an SN is assumed to be deployed on the deployment point equivalent to the gene's value and the coverage of the RoI is updated, i.e. the algorithm registers the target points that are covered thus far. This process is continued until full coverage is achieved. After the decoding process, the chromosome is given a fitness value equal to the number of SNs required for achieving full coverage of the set of designated target points in the RoI. The authors use the Order Mapped Crossover (OMX) scheme and a simple inversion mutation scheme to recombine and mutate chromosomes respectively. The algorithm follows a $(\mu + \lambda)$ scheme, in which a population of μ chromosomes is recombined to produce λ offspring. The fitness values of the $(\mu + \lambda)$ chromosomes are evaluated and the μ chromosomes with the highest fitness are kept for the following generation (i.e. iteration). Experimental results presented in [73] shows that the proposed iFLGA has a superior performance to the Greedy Heuristic (GH) MAX-AVG-COV proposed in [29] in terms of the quality of the obtained solutions to the studied problem.

In [74], the authors consider the problem of maximizing the area coverage of a 2-D flat RoI using a predefined number of heterogeneous SNs, with three different sensing ranges. The sensing model expressed in (2.1) is used. The authors propose the use of GA for solving this problem, which they term *Maximum Coverage* SDP (MC-SDP). Candidate solutions in the phenotype/solution space are converted to the genotype space using an integer-encoded chromosome similar to that used in [69] and [70], as expressed in (2.21). The only difference

is that the coordinates of the SNs which belong to the same SN type in a chromosome are contiguous, i.e. each chromosome is composed of three sections, each belonging to one type of SNs. According to this encoding scheme, the translation between the genotype and the phenotype spaces is not unique, i.e. there is inherent redundancy in the genotype space which arises from the fact that different permutations of each of the three sections of a chromosome are translated to the same candidate solution. The authors argue that this redundancy slows down the GA convergence and propose the use of a normalization technique [74] to overcome this problem. The authors use random parent selection in their proposed normalized GA, which is an odd choice since it is inferior in terms of convergence speed to the more commonly used Roulette Wheel or Tournament selection methods. In addition, the proposed algorithm shares the same limitations of the algorithm proposed in [69] and [70].

Similar to the studies in [24] and [73], the study in [75] also considers the MCC-SDP under the same assumptions and WSN model adopted in [24] and [73]. To find high quality solutions to the MCCDP, the authors propose an Integer-Encoded Variable Length Chromosome GA (iVLGA). This encoding scheme is used to enhance the computational efficiency of iFLGA proposed in [24] by avoiding the use of long redundant chromosomes, especially for large-scale instances of the MCC-SDP. Each chromosome, denoted by $c(l)$, contains l unique integer-valued genes, where $l \in [1, n]$, where n is the number of SN deployment points in the RoI. Each gene in the chromosome represents an SN deployed at a deployment point whose index is equivalent to the value of the gene. The following equation is used to evaluate the fitness of a chromosome $c(l)$, where $cov\big(c(l)\big)$ represents the number of target points covered by $c(l)$ and w is a constant:

$$f\big(c(l)\big) = -\Big(l + w * \Big(m - cov\big(c(l)\big)\Big)\Big), \tag{2.27}$$

The second term in (2.27) is responsible for penalizing solutions that do not provide full coverage of the m target points by assigning them a lower fitness value. The negative sign is added so that the maximum fitness would correspond to the deployment(s) achieving full coverage with the minimum number of SNs. The algorithm uses a special variant of the famous single-point crossover. A simple random mutation scheme is used in which the value of a gene is changed randomly and any repetitions of integers inside the mutated chromosome are discarded. For the selection schemes, the iVLGA uses a combination of the Roulette Wheel and Elitism selection schemes. Experimental results show that the proposed iVLGA in [75] outperforms the iFLGA in [73] in terms of both the quality of obtained solutions and the speed of convergence for all tested MCC-SDP instances. Results also suggest that in terms of scalability, iVLGA performs progressively better than iFLGA as the problem scale increases.

Similar to [74], the study in [76] addresses the MC-SDP. They adopt the same set of assumptions as in [74] with the exception of the SN heterogeneity assumption. To solve the MCSDP, they propose a GA with real-number encoded chromosomes. The chromosome encoding scheme adopted in the proposed GA is the same as the one proposed in[69] and expressed in (2.21). The fitness function of the proposed GA, denoted by $f(x)$, is given by:

$$f(x) = \sum_{i=1}^{n} C(s_i, A) - \sum_{i=1}^{n} \sum_{j=i+1}^{n} I(s_i, s_j), \tag{2.28}$$

where the term $\sum_{i=1}^{n} C(s_i, A)$ represents the total coverage of the RoI of the SN deployment represented by the chromosome. The term $\sum_{i=1}^{n} \sum_{j=i+1}^{n} I(s_i, s_j)$ represents the total area of the RoI covered by two SNs. The authors justify the design of the fitness function by the

following reasoning. Penalizing the coverage overlap between SNs will increase the speed of convergence of the proposed GA to solutions of good quality. The authors adopt uniform crossover and single point random mutation for the genetic operators. Tournament selection and a mixture of elitism and fitness-based selection are adopted for parent selection and survivor selection, respectively. Results suggest that the proposed GA is capable of significantly improving the RoI coverage of an initial random deployment of a predefined number of SNs. However, the authors did not provide evidence that their proposed GA has any advantage over similar existing algorithms. The authors also neglected the overlapped coverage area among more than two SN in their fitness function design. In addition, the proposed algorithm shares the same limitations of the algorithm proposed in [69] and [70].

To conclude this sub section, we present a comparison between the reviewed deployment algorithms which were based on the genetic approach in Table 2.4. The comparison is carried out in terms of the GA type, objective(s), encoding method, type of SNs deployed and the adopted sensing model.

Table 2.4 Comparison among the GAs designed for WSN deployment

Algorithm	Type of GA	Encoding	Objective(s)	Type of SNs	Sensing Model
in [69]	standard, multi-objective	real-number	• maximize coverage • maximize lifetime	homogeneous	deterministic
in[70]	standard, multi-objective	real-number	• maximize coverage • maximize survivability • minimize no. of SNs	homogeneous	deterministic
in [24]	microbial	binary	• cover a set of target points • minimize no. of SNs	heterogeneous ; acoustic and image	deterministic
in [72]	microbial	binary	• cover a set of target points • minimize no. of SNs	homogeneous; Infrared	probabilistic
in [73]	standard	integer	• cover a set of target points • minimize no. of SNs	homogeneous	deterministic
in [74]	normalized	integer	• maximize coverage	heterogeneous	deterministic
in [75]	variable-length chromosome	integer	• cover a set of target points • minimize no. of SNs	heterogeneous	deterministic
in [76]	standard	real-number	• maximize coverage	homogeneous	deterministic

2.4.2. Computational Geometry-based Algorithms

As explained earlier in Section 2.3.2, the VD and DT are famous CG structures that have been linked to WSNs deployment. Their unique properties were proven useful in evaluating area coverage and detection of coverage holes. In this section we review examples of the proposed deployment algorithms in literature utilizing these CG structures.

In [77], the authors utilize the VD in the *re*-deployment of a MWSN. They assume that the MWSN consisted of both static and mobile SNs to reduce the cost of deployment. Their deployment protocol, called *Bidding Protocol* is distributed; it is carried out by SNs concurrently and iteratively. The mobile SNs in the network are treated as *servers* that are used to *heal* coverage holes detected by static SNs based on their locally constructed Voronoi cells, assuming the binary sensing model in (2.1). In every protocol iteration, each mobile SN is assigned a *base price*, which is proportional to the size of the coverage hole it would leave behind if it moved to another location. Static SNs with detected holes in their Voronoi cells estimate the size of the hole and the candidate position for a mobile SN to move to in order fill that hole. This information is broadcast by the static SN in the form of a single parameter called a *bid*, which is essentially proportional to the size of the coverage hole detected. A mobile SN node receiving multiple bids chooses the highest one and relocates to heal the biggest coverage hole, provided that the bid is higher than its base price. The protocol terminates when there are no more bids broadcasted in the network higher than the base prices of the mobile SNs.

Results presented in [77] demonstrate the effectiveness of the Bidding Protocol in terms of coverage, deployment cost and speed of convergence. In terms of the number of SNs required to achieve certain coverage of the RoI, the presented protocol requires a significantly smaller number of SNs, with the reduction reaching as high as 50%, when compared to a random deployment. In terms of the speed of convergence, the protocol terminates after six iterations or less for different mobile to static SNs ratios. This is advantageous since it translates to less energy expenditure and hence a higher network's lifetime. However, the drawback of the Bidding Protocol is that it deals only with a RoI with a flat terrain with no obstacles. This is an unrealistic assumption; obstacles can prevent static SNs from constructing their Voronoi cells correctly (since this is carried out through detecting neighbors using wireless broadcast) and hence the coverage holes detection mechanism would be inaccurate. Also, obstacles can obstruct the movement of mobile SNs to their target locations.

The authors in [48] consider the problem of re-deploying an initially randomly deployed MWSN, consisting of n identical mobile SNs of the same communication and sensing ranges, to maximize area coverage. The authors assume a 2-D RoI with a flat terrain and wall-like boundaries, and adopted the binary sensing model in (2.1). The authors propose a distributed and iterative deployment algorithm for MWSNs called Minimax. The algorithm also depends on locally constructed Voronoi cells as in [77]. The target location for SNs in each iteration in Minimax is a point inside the local Voronoi cell called the Minimax point. The location of the Minimax point is chosen such that the variance of the distances between SNs and their Voronoi vertices is minimized, resulting in a more evenly shaped Voronoi cells and hence a more even distribution of SNs in the RoI. This in turn leads to maximizing the area coverage of the MWSN, since each SN can cover its own local Voronoi cell more effectively. The authors used analytical derivation to prove their proposed method for computing the Minimax point. The computational complexity of the algorithm is $O(m^3)$, where m is the number of the Voronoi vertices belonging to specific SN. The results presented suggest that the algorithm is capable of achieving its objective of maximizing the area coverage of an initially randomly deployed MWSN. However, Minimax suffers from two drawbacks. The first one is the same drawback of the bidding protocol in [77]; the performance of both algorithms would possibly deteriorate in the presence of obstacles, since the locally constructed Voronoi cells can be inaccurate in this case. The second one is the high energy expenditure; SNs move after the conclusion of each iteration, which would certainly decrease the network's lifetime.

In [50], two distributed deployment algorithms are proposed with the objective of maximizing the area coverage of MWSNs, namely the *Centroid* and the *Dual-Centroid*

algorithms. Both algorithms are distributed and iterative. The termination condition can be either reaching the predetermined maximum number of iterations or reaching the required coverage level. To compute the level of area coverage reached in each iteration, the authors use the locally constructed Voronoi cells of the deployed SNs in the same manner as in [77] and [48]. Inspired by [78], the proposed Centroid scheme is a simple algorithm that makes use of the geometric center, or the *centroid*, of the Voronoi cells. The centroid of any polygon is defined as the intersection of all the lines that divide the polygon into two equal parts. Fig. 2.4 provides an illustration of the definition on a triangular polygon XYZ, where point *c* is its centroid. After the initial random deployment, SNs establish their local Voronoi cells concurrently. Each SN node would then calculate the coordinates of the centroid of its cell (C_1) and checks if its local coverage would increase if it moves from its current location to C_1. This step implies an assumption that the communication range r_c is greater than the sensing range r_s , which is, for many types of SNs, a correct assumption.

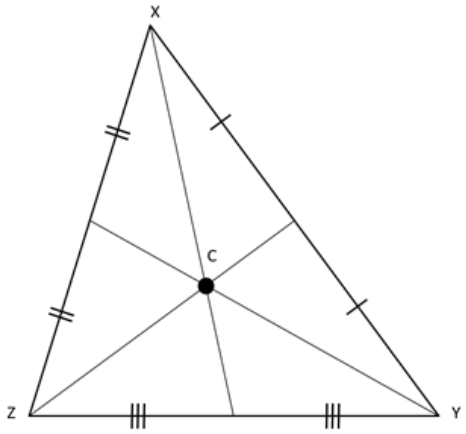


Fig. 2.4 Centroid, c, of triangle $\angle XYZ$

This constitutes a single iteration. The Dual-Centroid algorithm utilizes two centroids, the Voronoi cell's centroid C_1 and the centroid of the *Neighbor Voronoi cell*, C_2. The Neighbor Voronoi Cell, C_2, is the polygon with the SN's neighbors as vertices. The same steps carried out in the Centroid scheme are used, except that the new position for a SN *G* is calculated using the following equation:

$$G = \alpha C_1 + (1 - \alpha) C_2 \qquad (\alpha \in (0,1)) \,, \tag{2.29}$$

The authors compare the performance of both algorithms to Minimax in [48]. Results presented suggest that both algorithms have a better coverage performance compared to Minimax, with the Dual-Centroid algorithm being the best. This slight advantage in coverage performance is accompanied by a higher computational complexity, since two Voronoi cells and their centroids must be computed. It should be noted that although the coverage performance was enhanced in [50], the authors didn't address the drawbacks of [48].

The authors in [79] consider the same re-deployment problem as in [77], [48] and [50], but with energy consumption in mind. They presented the Voronoi Diagram Deployment Algorithm, or VDDA, to solve the problem in a distributed fashion, with the same assumptions used in [50]. VDDA is similar in its steps to the Centroid scheme in [50]. However, VDDA considers multiple points for the next position of the SN node i.e. conducts a local search inside each polygon for the optimum candidate position for relocation. One of

these positions is the centroid of the SN's Voronoi cell. Another candidate position is the center of the Voronoi cell *range*, which is defined as the midpoint of the maximum and minimum points along the x and y coordinates in the Voronoi cell. The search space is reduced to several points that are linearly spaced, starting from the current position of the SN to the centroid and then to the center of the Voronoi cell range. These points are evaluated before the actual movement takes place. The evaluation is based on what the authors called a node utility metric U_t^i given by:

$$U_t^i = A_t^i \times T_t^i \,, \tag{2.30}$$

where A_t^i is the effective coverage of SN i at iteration t, while T_t^i represents the estimated lifetime of SN i at iteration t after the projected movement from the SN's current position to the candidate position. The lifetime parameter T_t^i is calculated by assuming an initial equal amount of energy for each SN and setting a specific value for energy consumption per unit distance travelled by the SNs. After finding the best point in the search space, according to that metric, the SN moves to that position. From the presented results, VDDA succeeds in increasing the initial coverage of the random deployment and results in an almost uniform distribution of SNs in the RoI. The authors argue that this advantage is of great importance, since uniformity of deployment can decrease interference between SNs and consequently decrease communication energy consumption, increasing the network's lifetime. However, the algorithm is slow and complex, increasing the processing load on each SN. VDDA also does not address the presence of obstacles in the RoI.

On the other hand, DT was utilized in a centralized deployment algorithm proposed in [80]. The deployment algorithm, called *DT-Score*, is tailored for WSN applications that involve a RoI containing stationary obstacles in known locations prior to the deployment (by the means of satellite imagery, for example). The objective of DT-Score is to maximize the area coverage of the RoI using a fixed number of static SNs n. All SNs have identical communication range r_c and a probabilistic sensing model as expressed in (2.2).

DT-Score algorithm runs in two phases, the *contour* deployment and the *refined* deployment respectively. In contour deployment phase, SNs are deployed at regular intervals along the edges of the RoI (assumed to be rectangular shaped) and the obstacles (assumed to be polygon shaped). This phase is used to eliminate coverage holes at the boundaries of the RoI and the obstacles. In the refined deployment phase, a sequence of steps is repeated, where each repetition results in the deployment of one more SN in the RoI. These steps consider the set of deployed SNs so far and locate existing coverage holes using DT. The empty circle property, discussed earlier in Section 2.3.2, is used to locate candidate positions for new SNs. These candidate positions are simply the centers of the empty circles generated from the DT of the set of deployed SNs. The candidate positions are then scored according to the coverage gains they would produce (based on both the radius of the empty circle and the vicinity to obstacles). At the conclusion of each repetition, a SN is deployed in the candidate position of the highest score. The second phase is concluded when the predetermined number of deployable SNs n is reached.

As a centralized algorithm, the DT-Score algorithm has a key advantage over other centralized grid-based algorithms, such as the MAX-MIN-COV in [29], which is its scalability. The computational complexity of DT-Score is $O(n^2 logn)$, whereas most grid-based algorithms have a complexity is $O(N^2)$, where N the number of grid points in the RoI is. Although DT-Score considers the presence of obstacles in the RoI, its limitations lie in the assumption that the exact topology of the RoI is known prior to the deployment. This assumption is not applicable for some WSNs applications.

In Table 2.5, we present a comparison between the reviewed algorithms in this section in terms of the CG structure used, the type of deployed SNs (static only; mobile only; or both), the RoI's terrain (flat or containing obstacles), adopted sensing model and whether the algorithm is centralized or distributed.

Table 2.5 Comparison among the CG-based algorithms for WSN deployment

Algorithm	CG Structure	Type of SNs	ROI	Sensing Model	Centralized/Distributed
Bidding protocol in [77]	VD	heterogeneous; static and mobile	w/o obstacles	deterministic	distributed
Minimax in[48]	VD	homogeneous; mobile	w/o obstacles	deterministic	distributed
Centroid and Dual Centroid in [50]	VD	homogeneous; mobile	w/o obstacles	deterministic	distributed
VDDA in [79]	VD	homogeneous; mobile	w/o obstacles	deterministic	distributed
DT-Score in [80]	DT	homogeneous; static	w/ obstacles	probabilistic	centralized

2.4.3. Artificial Potential Field-based Algorithms

A large number of studies presented deployment algorithms that utilize the concept of Artificial Potential Field (APF), which is also referred to as Virtual Forces (VF) in the literature. Since the concept of APF depends on *motion*, deployment algorithms based on APF are usually re-deployment algorithms for MWSNs. However, they can also be used in the planned deployment of static WSNs through simulation. These algorithms can be either be centralized or distributed. In the following two sub-sections, we review APF-based algorithms proposed in each type, along with its advantages and disadvantages.

2.4.3.1. Distributed Algorithms

Distributed APF-based deployment approaches are used in de-centralized MWSN architectures, where every SN uses its local data, such as distances to neighboring SNs and obstacles, to run the deployment algorithm and self-deploy in the optimized positions. There are two advantages to this approach. The first advantage is that it doesn't depend on any prior knowledge of the RoI. It also doesn't require any coordination between the SNs, as they are assumed to run the algorithm concurrently. Hence the distributed algorithms are usually highly scalable, i.e. the algorithm's complexity or performance isn't sensitive to the number of deployed SNs.

On the other hand, applying this approach involve extending the conventional sensing capabilities of SNs in order to be able to obtain the required type of local data. For example, SNs need to be equipped with laser range finders to detect obstacles and neighboring SNs. It also involves extending their computational power to carry out the distributed algorithm, which compounded with the actual SN movement, can increase the power consumption levels significantly and consequently decrease the MWSN's lifetime.

One of the earliest proposed distributed deployment algorithms based on APF is the algorithm presented in[60], which considers the problem of deploying a MWSN in an unknown environment that may be dynamic and even hostile. The authors propose an APF-based approach for deployment, in which the mobile SNs are treated as virtual free particles

that are subject to virtual forces. These forces repel the SNs from each other and from obstacles, i.e. the artificial potential field in (2.7) is caused by obstacles and other SNs in the network, and its gradient results in two virtual repulsive forces, as expressed in (2.8) - (2.11). This method guarantees that an initial compact configuration of SNs will spread out to maximize the area coverage of the MWSN in a certain RoI. In addition to these repulsive forces, SNs are also subject to a viscous friction force. This force is used to ensure that the network will eventually reach a state of static equilibrium; i.e. nodes will eventually come to a complete stop, given that the RoI itself eventually becomes static. The proposed algorithm computes the trajectory of the mobile SN by applying the following equation of motion:

$$\ddot{x} = (F - \gamma \dot{x})/m \, , \tag{2.31}$$

where $\boldsymbol{F}$ is the resultant force vector on a given SN, $\ddot{\boldsymbol{x}}$ and $\dot{\boldsymbol{x}}$ are the acceleration and velocity vectors respectively, γ is the viscosity coefficient and m is the SN's mass. This virtual equation of motion assumes the SN is a free particle, and hence it must be mapped to a real control law. This control law takes into account both the kinematic and dynamic constraints of a mobile SN, such as its maximum velocity and acceleration. The simulation results offered in this paper suggest that the proposed algorithm has the potential of introducing a huge improvement that reaches 10-fold in terms of RoI coverage from an initial compact deployment of SNs. The deployment time was measured and found to be considerably fast, given the limited maximum velocity assumed for the mobile SNs (0.5 m/s). An unplanned but appealing feature of the final deployment was observed, which was the evenness of the inter-distance between the SNs. The authors do not offer a solid explanation for this phenomenon. They also do not study the effect of the deployment algorithm on the energy reservoir of the nodes.

The authors in [81] propose a similar APF-based deployment algorithm. They consider SN-carrying robots that are deployed in a certain RoI, aiming to achieve certain goal. The goal is to detect a certain phenomenon occurring in the RoI by at least four of the deployed robotic SNs and communicating data successfully to a fixed sink node. The proposed algorithm computes the resultant force vector $\boldsymbol{F}$ acting on each SN, and applies the equation of motion expressed in (2.30) to direct the SN, as in [60]. However, the authors in[81] assume that the virtual forces acting on each SN are both repulsive and attractive, as opposed to only repulsive in [60]. The repulsive forces in the algorithm are exerted by the obstacles in the RoI and other robotic SNs. There are three types of attractive forces considered in the algorithm. The first type include attractive forces caused by the goals in the RoI, while the second type include attractive restoring forces based on penalties for exceeding the maximum allowable communication range r_c between SNs. The third type includes attractive forces based on maximizing the capacity between nodes, and it depends on the optimal value of a communication utility function proposed by the authors. This communication utility function aims to maximize the use of the WSN capacity by adjusting the sources' rates to their optimal values. Since the SNs are mobile, it is hence dependent on their locations. This is because the maximum achievable data rate between any two nodes depends on the distance between them. This type of attractive force acting on a certain SN i with a location vector $\boldsymbol{r_i}$ is given by:

$$F_i = -\frac{\partial U^*}{\partial \boldsymbol{r}_i}, \tag{2.32}$$

where $U^*$is the optimal value of the communication utility function. The algorithm is tested on a scenario consisting of a 2-D rectangular RoI with a single target at a specific location and a sink node in another fixed location. Although the algorithm factored in the presence of

obstacles, no obstacles were included in the RoI in the simulation. The results presented showed that the algorithm successfully enabled the MWSN to attain its target in the considered scenario. However, we find that this scenario is unrealistic since it is unlikely that the exact position of the phenomenon or target of interest is known in advance.

In [48], the authors present two APF-based distributed deployment algorithms for MWSNs. These algorithms aim to increase the area coverage of a defined RoI in which mobile SNs were initially randomly deployed. The algorithms run concurrently and iteratively until the desired area coverage is reached or the maximum numbers of runs is executed. It is assumed that all deployed mobile SNs are identical with a fixed sensing range r_s. Both algorithms utilize the Voronoi diagram; in the beginning of each algorithm, all deployed SNs construct their Voronoi cells based on their relative position from neighboring SNs and the wall-like boundaries of the RoI. This procedure was explained in Section 2.4.2 and was part of the algorithms presented in [50], [77] and [79].

In the first algorithm, named VEC (VECtor-based), both repulsive and attractive forces are exerted on each mobile SN in the network. These positive and negative forces take place in between SNs and also between SNs and the wall-like boundaries of the RoI, which are considered obstacles exerting repulsive forces on nearby SNs. The algorithm depends on the following idea: for complete area coverage of the RoI, SNs should be uniformly deployed such that the inter-distances between them are constant and equal to d_{ave} (or $d_{ave}/2$ between a SN and RoI boundary) which is calculated according to the area of the RoI, sensing range r_s and the number of deployed SNs. If the distance between two SNs is less than d_{ave}, ($d_{ave}/2$ between a SN and RoI boundary), a virtual repulsive force acts on the SNs to increase the separation between them to d_{ave} , and vice versa. In each iteration, each SN computes the resultant force vector $\boldsymbol{F}$ acting on it and determines its next target position. To reduce errors, the authors added a movement-adjustment scheme which allows SN movement in each round only if the local coverage would be enhanced by such movement. The local coverage of a SN is simply the intersection between its Voronoi cell and its circular sensing area with a radius r_s .

The second algorithm in [48], named VOR (VORonoi-based), is a pull-based algorithm; only an attraction force is considered. In VOR, if a SN detects the existence of coverage hole in its Voronoi cell, it moves towards its farthest Voronoi vertex, such that the Euclidian distance between the target position and the farthest vertex becomes equal to the sensing range r_s. This is illustrated in Fig. 2.5, where vertex v_2 is the farthest vertex of the Voronoi cell of SN s_i and TP is the target position. The authors report that VOR resulted in moving oscillations due to its greedy nature in fixing the largest coverage holes. To deal with this problem, they added oscillation control to the basic algorithm, which prevents a SN from moving in opposite directions in two consecutive rounds. They also limit the maximum moving distance in VOR to only half the communication range r_c to deal with inaccurate construction of Voronoi cells due to communication limitations.

The authors carried out extensive experimentation on both algorithms to qualify their performance in terms of the coverage obtained, the accumulated moving distance by the SNs, scalability, and impact of the initial topology. Results show that in terms of total moving distance, VEC was the more efficient. In terms of scalability, both algorithms proved to be extensible to large deployment scenarios, since the communication and movement are kept local in these algorithms. In terms of the impact of the initial topology, the authors consider

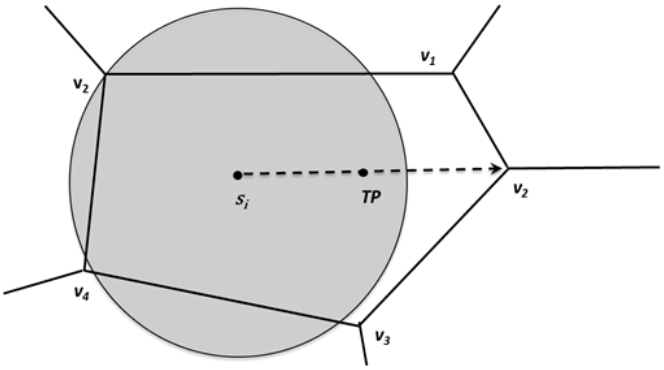


Fig. 2.5 VOR algorithm proposed in [48]

both a random and a normal initial topology. The results show that the proposed algorithms take more time to converge in the case of the normal distribution than in the random case. However, the apparent drawback of both algorithms is not factoring in the presence of obstacles in the RoI. As we pointed earlier, the presence of obstacles may result in inaccurate Voronoi cells. Another drawback is the repeated movement of SNs, reducing the network's lifetime, although the movement is reduced to an extent in VOR, since a SN is only allowed to move if its local coverage will be enhanced.

In [49], the authors consider the same re-deployment problem as in [48] and use the same assumptions. Voronoi cells, constructed by SNs using local data, are utilized in the proposed algorithm, in order to divide the RoI into parts so that each SN can maximize its local coverage. The proposed algorithm combines repulsive forces from neighboring SNs and the RoI boundaries, and the attractive force that draws the SN to the centroid of its Voronoi cell. In each iteration of the algorithm, each SN calculates its local Voronoi cell. If no coverage hole is detected, the SN would not calculate the centroid of the cell and would just consider virtual forces from neighboring SNs and possibly the RoI boundaries. Otherwise, the SN calculates the centroid of its Voronoi cell. The resultant force vector $\boldsymbol{F}$ is computed accordingly, and the SN's trajectory is computed by applying the equation of motion expressed in (2.31). As in VEC algorithm proposed in [48], the SN only moves to a new location if its local coverage will be enhanced. The algorithm is terminated when each of the deployed SNs reaches an equilibrium state by itself. The equilibrium state, in turn, can occur in two cases; if the SN repeatedly moves back and forth to the same position (oscillatory equilibrium state) or if its accumulated moving distance during a given time period set in the simulator does not exceed a certain threshold value (stationary equilibrium state).

The authors compare their algorithm to the one proposed in [60] in terms of coverage, cumulative moving distance and deployment time. Results suggest the proposed algorithm in [49] shows a better coverage performance than the algorithm in [60], irrespective of the number of SNs. The performance gain diminished as the number of deployed SNs increased. This was attributed to the fact that a relatively large number of deployed SNs can easily cover the RoI adequately without much need of a complex self-deployment algorithm. The proposed algorithm also exhibited superior performance in terms of the cumulative moving distance and area coverage. The authors attribute this to the additional attractive force that optimizes the SNs' paths in terms of maximizing the coverage. They also point out that the algorithm is sensitive to the communication range r_c, assuming a constant number of deployed SNs and area of the RoI. This is attributed to the fact the proposed algorithm depends on the correct estimation of the Voronoi cell of each SN. If the SN cannot detect its neighbors due to a relatively short r_c, it will estimate an inaccurate Voronoi cell. Again, we point out that this error can also take place in a RoI containing obstacles, which were not considered in [49]. Hence, the comparison between the performances of the APF-based

algorithms in [60] and [49] may be inaccurate since authors in [60] considered a complex RoI with obstacles such as walls and doors.

In [82], the authors propose a comprehensive distributed APF-based algorithm, named Holes dEtection and healing (HEAL), for mobile WSN re-deployment. HEAL is designed to detect, measure and fix coverage holes in MWSNs that arise due to random deployment, environmental factors or external attacks on the network. The algorithm operates in two phases. In the first phase, a distributed hole detection algorithm is used to find coverage holes in the RoI, estimate their size and locate their center, using the Gabriel Graph (GG) [46] of the WSN. In the second phase, the mobility of the SNs is exploited to cover the detected holes. This is carried out through applying an APF-based distributed relocation algorithm, where SNs in the vicinity of a hole are subject to an attractive virtual force exerted by its center, in addition to repulsive virtual forces in between them to minimize coverage overlap. To validate the algorithm, the authors applied HEAL to different scenarios by varying the number and sizes of the holes. Results indicate that the proposed algorithm has the ability to almost perfectly detect and repair coverage holes in densely deployed MWSNs. However, the algorithm can only deal with obstacle-free RoIs and does not address holes which occur on the borders of the RoI.

Similar to [81], the study in [83] considers the problem of the re-deployment of mobile SNs which are initially randomly deployed in RoI. The objective is to achieve the coverage of single or multiple target locations while maintaining connectivity with the sink node in the presence of potential obstacles in the RoI. The authors propose an APF-based deployment algorithm which is coined the Obstacle Avoidance Target Involved Deployment Algorithm (OATIDA). The proposed algorithm runs concurrently on each SN in iterations where it calculates the SN movement vector $\overrightarrow{M_i}$, where i is the index of the SN. The algorithm starts by constructing the Relative Neighborhood Graph (RNG) using the neighbors' relative distance and indicating the distance to the farthest SN in the constructed RNG. This distance is used to calculate the *magnitude* of the distance that will be travelled by the SN in the current iteration, which is denoted by D_i. The preferred *direction* of motion for the SN in the current iteration, denoted by $\overrightarrow{D_i}$, is then decided using the advertised target(s) location. Following this step, the algorithm computes the repulsive force, denoted by $\overrightarrow{FO_i}$, which is exerted by the nearby obstacle(s) in the RoI on the SN. Finally, the movement vector $\overrightarrow{M_i}$ of the current iteration for SN i is calculated using the following equation:

$$\overrightarrow{M_i} = D_i . \overrightarrow{D_i} + \overrightarrow{FO_i} \tag{2.33}$$

Results presented in [83] shows that the proposed OATIDA is capable of directing the mobile SNs to successfully provide coverage for single and multiple targets in the presence of one or more obstacles while maintaining connectivity with the sink node. However, the speed of convergence of the algorithm cannot be commented on since the authors did not provide a performance comparison between their proposed algorithm and similar existing algorithms in the literature (e.g., [81]). The main drawback of the OATIDA is that it is based on the assumption that all the mobile SNs are initially deployed within the communication range of the sink node. This assumption is not realistic since, in most practical application of MWSNs, the initial random deployment cannot be precisely controlled.

All the previous examples considered homogeneous MWSNs, where all the deployed SNs are mobile. However, a MWSN can be heterogeneous in the sense that only a subset of all deployed SNs are mobile in order to decrease the network's energy consumption and cost while enhancing the coverage or any other performance metric. In [84], the authors propose an APF-based deployment algorithm that only affects the sink nodes in a randomly deployed

WSN, i.e. only sink nodes are mobile, to enhance the total coverage. The authors differentiate between the attractive and repulsive virtual forces between the sink nodes and the static SNs, the virtual repulsive forces in-between sink nodes, and the virtual repulsive forces on sink nodes due to the wall-like boundaries of the RoI. It is assumed that the sink nodes are SNs with higher energy reservoir, locomotive capabilities and higher sensing and communication ranges. The direction and magnitude of the virtual forces between static SNs and sink nodes depend on their interspacing, and aims at adjusting it to be equal to $\sqrt{3}r_s$, which is the interspacing in a regular hexagonal grid. The repulsive forces in between sink nodes are introduced to guarantee that the minimum distance between two sink nodes is their communication range in order to disperse the sink nodes on the RoI.

The simulation results presented are not extensive; they only provide an illustration on how the coverage of an initially randomly deployed WSN, confined to a rectangular RoI, was improved when about 15% of its nodes (the sink nodes) moved according to the proposed algorithm until reaching equilibrium. The authors do not specify what equilibrium in the simulation environment implies (static or oscillatory) and do not provide data on the effect of increasing or decreasing the introduced percentage of sink nodes in the WSN on its coverage.

2.4.3.2. Centralized Algorithms

APF-based centralized deployment algorithms are used in cluster-based WSN architecture, where the cluster head is assumed to have a high computational power to carry out the deployment algorithm for all deployed SNs. To carry out this task, the cluster head has to first localize the SNs in the network after an initial deployment and collect any other data pertinent to the deployment algorithm. After running the algorithm, the cluster head then communicate to each SN its new target position.

The advantages of the centralized approach lie in the fact that only one (or a few) resourceful cluster-head is responsible for running the deployment algorithm. The deployed mobile SNs are not required to possess any extra sensing or computational abilities, apart from the ones required for their primary sensing function. However, they are required to have the ability of self-localization, in order to make sense of the communicated target locations. The centralized approach also requires that all deployed SNs can communicate with the cluster head. It is readily apparent that this kind of approach is not feasible in dynamic and harsh RoIs, such as disaster areas and battlefields. In such cases, ensuring the survival of a resourceful server is very difficult. This approach also requires a prior knowledge of the terrain of the RoI, in order to be able to keep the mobile SNs away from obstacles. Consequently, if the RoI has a dynamic rather than a static nature, a centralized deployment approach would be infeasible.

An example of a centralized APF-based deployment algorithm deployment for MWSNs is proposed in [85]. The proposed algorithm, called Virtual Forces Algorithm (VFA), aims to maximize the coverage of a cluster-based MWSN, with a fixed number of deployed mobile SNs that are initially randomly deployed. A powerful cluster head is proposed to be responsible for carrying out the VFA, assuming it possesses augmented computational power over SNs. The VFA considers a combination of virtual attractive and repulsive forces on each SN in the RoI, due to neighboring SNs, obstacles and preferential areas. The resultant force vector F on each SN is used to determine its virtual target location in each iteration. The VFA terminates either when the required level of area coverage is achieved or when a predetermined number of iterations are reached. Once the VFA is concluded, the final target positions are identified and communicated to SNs, and a one-time movement is carried out.

The authors consider three scenarios to evaluate the performance of VFA. The first scenario an ideal one, assuming the binary sensing model as expressed in (2.1) and no obstacles or preferential areas. The second scenario assume a probabilistic sensing model as

expressed in (2.2), but again with no obstacles or preferential areas, while the third one assume a binary sensing model with a single obstacle and preferential area. Results show a substantial improvement in the area coverage of the RoI compare to the initial random deployment, with an almost constant interspacing between SNs. However, the VFA depends primarily on the assumptions that a prior knowledge of the RoI terrain (obstacles and preferential areas) is available and that the cluster-head is not threatened by energy depletion at any point during the VFA run time. Also, the initial random deployment is assumed to be at least $1-$connected; every SN can communicate with the cluster head in a single or multi-hops. This may not be the case in random deployments. Another limitation for VFA is its scalability, since the communication overhead would dramatically increase as the number of deployed SNs increase.

Another example is presented in [86]. The authors propose a centralized APF-based deployment algorithm called Target Involved Virtual Force Algorithm (TIVFA), which is executed on the cluster-head of an initially randomly deployed MWSN. The authors work on the following assumptions: in a well-defined RoI, there exist hotspot areas, obstacles, static target areas and maneuvering targets. The hotspot areas are defined as areas known to be more important in the MWSN, such as a headquarters in a battlefield. Static target areas are defined as circular areas in the RoI where targets are more likely to appear (depending on prior information). Maneuvering targets were divided into several importance levels, so that targets of higher importance level are given more attention in the WMSN. All SNs are assumed to be identical with a fixed communication range, self-localization capability and a probabilistic sensing model as expressed in (2.2). The objective of TIVFA is to reconfigure the area coverage after an initial random deployment, such that targets of higher importance are detected more precisely, while ensuring obstacle-avoidance and high-coverage of the hotspot areas. The algorithm, running on the cluster head, computes the resultant force vector $\boldsymbol{F}$ that constitutes both attractive and repulsive forces. The forces between SNs can be attractive or repulsive depending on the distance between them as in [85]. The hotspots and static target areas are assumed to exert attractive forces on SNs, depending on the distance between them and the radii of the hotspot and static target areas. The obstacles logically exert repulsive forces on SNs, also depending on the distance between them. Finally, the forces exerted by the maneuvering targets on the SNs are explained as follows: if a SN detects a target with a probability less than *1*, the target is assumed to exert an attractive force with a strength that is inversely proportional with the detection probability. Otherwise, no force is exerted on the SN by the target. The simulation results presented in [86] suggest that TIVFA succeeded in fulfilling its objectives. However, the algorithm depends primarily on prior knowledge of the RoI. It also implicitly assumes that the cluster head maintains the locations of all the SNs in the network as they move based on its instructions after each iteration or time interval. This will consequently introduce a considerable amount of traffic into the network. The combination of movement and communication overhead of TIVFA can diminish the network's lifetime significantly. Also, the authors do not offer explicit algorithm termination criteria for TIVFA, which is essential to understand the TIVFA's performance regarding energy consumption.

In [87], the authors propose a modification to the VFA algorithm presented in [85] to enhance its performance. They claim that the VFA produces coverage holes in the RoI upon its convergence. They attributed this phenomenon to the fact that in VFA, virtual attractive and repulsive forces due to all neighboring SNs, i.e. all SNs within the communication range of a SN, are considered. These forces can in some instances cancel out and prevent SNs from covering some areas in the RoI. In order to overcome this phenomenon, the authors propose a modified version of VFA that only takes into account *adjacent SNs* based on the DT of the initial random deployment. If two SNs are "connected" in the DT (were the set of points of

the DT are the initially randomly deployed SNs), then they are adjacent SNs. Results presented in [87] suggest that the modified VFA is superior to the original VFA in [85] both in terms of the total coverage of the RoI and its uniformity. A comparison between the processing times of both algorithms is also presented and it shows that the modification proposed does not add any significant processing cost. However, similar to the original VFA, the modified VFA will not function correctly if some of the SNs in the initial random deployment are unable to communicate with the cluster head.

We conclude this section by comparing between the APF-based algorithms reviewed above in Table 2.6. We compare between them in terms of the repulsive and attractive virtual forces exerted on SNs, their objectives, the sensing model adopted and whether the algorithm is centralized or distributed.

2.4.4. Swarm Intelligence Algorithms

The use of SI methods in the deployment of WSNs is relatively new and limited, compared to the other three approaches discussed in the previous sections. To the best of our knowledge, the first research effort in this approach was presented in [88]. The authors study the problem of deploying a finite number N of homogeneous mobile SNs to cover a 2-D RoI. The SNs are assumed to be initially randomly deployed, and a proposed PSO-based deployment algorithm, called PSO-Grid, is assumed to run on a computationally powerful base station. The base station would send the optimized positions to the randomly deployed SNs after the convergence of the algorithm to maximize the area coverage of the RoI. Each particle in the swarm used in PSO-Grid represents a deployment solution that contains the positions of all the SN nodes inside the RoI. Hence, PSO-Grid encodes each particle in the swarm as follows: the position of a single SN j is described by its Cartesian coordinates (x_j, y_j), and for N SNs, the dimension of a particle in the swarm is two times the number of SNs, i.e. $2N$. This is similar to the encoding used in [69] and expressed in (2.21). The algorithm starts by randomly generating a number of solutions or particles. Equation (2.17) is used in PSO-Grid with a minor refinement of multiplying the current velocity term by an inertia weight w, which is used to control the effect of the previous velocity in the current velocity. A time decreasing inertia weight encourages high exploration of the search space at the beginning and fine tunes it at the end, as suggested in [65]. Hence, (2.17) is modified in the algorithm as follows:

$$v_{ij}(t+1) = w\, v_{ij}(t) + c_1 R_1 \left(m_{ij}(t) - p_{ij}(t)\right) + c_2 R_2 (m_{gj}(t) - p_{ij}(t)) \,, \qquad (2.34)$$

where i is the index of the particle in the swarm and j is the dimension of the particle, hence $j = 1,2, \ldots 2N$. In order to evaluate each solution, i.e. the current position of a particle in the search space, a fitness function is used. Logically, the fitness function chosen in PSO-Grid is the area coverage of the RoI. Calculating the area coverage is carried out by creating a uniform grid over the RoI. All the grid points located in the RoI is labeled 1 or 0, depending whether it is covered by at least one SN or not, assuming the binary sensing model as expressed in (2.1). The coverage is simply the ratio of the summation of ones to the total number of grid points. The authors evaluate the performance of PSO-Grid by comparing its results with a similarly encoded GA. The results suggest that although both algorithms converge to near optimum solutions after a maximum number of set iterations, PSO exhibits a significantly faster convergence than GA.

Table 2.6 Comparison among the APF algorithms for WSN deployment

Algorithm	Sources of repulsive VFs	Sources of Attractive VFs	Objectives	Sensing Model	Centralized/ Distributed
[60]	• obstacles • SNs	–	• maximize area coverage	not stated	distributed
[81]	• obstacles • SNs	• SNs • stationary targets • maximizing link capacities between SNs	• detection of stationary targets • maximize network's throughput	deterministic	distributed
VEC [48]	• ROI boundaries • SNs	• SNs	• maximize area coverage	deterministic	distributed
VOR [48]	–	• farthest vertex in local Voronoi cell	• maximize area coverage	deterministic	distributed
[49]	• ROI boundaries • SNs	• Centroid of local Voronoi cell	• maximize area coverage	deterministic	distributed
HEAL [82]	• SNs	• detected coverage holes	• maximize area coverage • minimizing SN movement	deterministic	distributed
OATIDA[83]	• obstacles	• stationary or moving targets	• coverage of designated targets	deterministic	distributed
[84]	• ROI boundaries • Static SNs • mobile sink nodes	• Static SNs • mobile sink nodes	• maximize area coverage	deterministic	distributed
VFA [85]	• ROI boundaries • obstacles • SNs	• SNs • preferential areas	• maximize area coverage	probabilistic	centralized
TIVFA[86]	• SNs • obstacles	• hotspots • static target areas • maneuvering targets • SNs	• maximize area coverage • detection of static and maneuvering targets	probabilistic	centralized
[87]	• ROI boundaries • SNs	• SNs	• maximize area coverage	deterministic	centralized

The study in [89] considers the generic problem of maximizing the area coverage of a WSN composed of a finite number of homogeneous static SNs in a 2-D RoI. To solve the problem, the authors combines PSO and Voronoi diagram to build a deployment algorithm that converges to the optimal positioning of the available SNs in terms of area coverage. The same particle encoding and time decreasing inertia weight in PSO-Grid [88] are used in [89]. However, the presented algorithm uses the Voronoi diagram instead of a grid to calculate the percentage of the covered area of the RoI for each candidate solution. After the conclusion of each iteration, SNs in their current positions generate their Voronoi cells. Only the distances between a SN and its Voronoi vertices are checked to ensure that the Voronoi cell is covered, assuming a binary sensing model. As for the RoI boundaries, a finite set of boundary points are selected at random and checked to be covered. Both the Voronoi vertices and the boundary points are referred to as *interest points*. Accordingly, the fitness function in the deployment algorithm depends solely on the distances between these *interest points* and their nearest SNs as shown in Fig. 2.6. The presented deployment algorithm is evaluated in several scenarios, where the effect of the number of deployed SNs and the size of the 2-D RoI on area total coverage is studied.

```
Interest points = [Voronoi cell vertices, n randomly selected points along the boundary];
For each interest point
Find the distance of the interest point to its nearest sensor;
If distance>sensing radius
Fitness += (distance−sensing radius);
End
```

Fig. 2.6 Fitness function in [89]

It is important to point out that the authors calculate the area coverage as the percentage of the interest points covered to the total number of interest points, similar to the approach used in [84]. The results show that although the presented algorithm followed the logical trends in both investigations (number of SNs and size of RoI versus area coverage), it produces sub-optimal results after a predetermined maximum number of iterations.

In [90], the authors extend their work by refining their deployment algorithm presented in [89]. The refined algorithm is named PSO-Voronoi, and it solves a similar deployment problem to the one in [89] and [88]. However, the fitness function in PSO-Voronoi was changed to be the total area of the coverage holes in the RoI for increased accuracy. To calculate the area of coverage holes, the authors again use the Voronoi diagram and the notion of *interest points*, as defined in[89]. The calculations are straight forward; if the distance d between an interest point and its nearest SN is greater than the sensing radius r_s, then a coverage hole exists around the interest point. The area of the hole is approximated to quarter, half or a full circle, with a radius $d - r_s$, according to the location of the interest point (corner, boundary or inside the RoI respectively). The authors propose that PSO-Voronoi be executed in a centralized manner: the algorithm is executed on a base station node, which would then communicate the coordinates of the optimized positions to randomly deployed mobile SNs.

The PSO-Voronoi algorithm's performance is compared to that of PSO-Grid algorithm presented in [88]. In terms of complexity, the comparison is in favor of PSO-Voronoi. This is because its complexity depends only on the number of interest points (or the number of SNs deployed), and not on the actual size of the RoI. On the other hand, the complexity of PSO-Grid depends on both the number of deployed SNs and the number of the grid points. The

energy consumption aspect of PSO-Voronoi is not discussed. However, it is obvious that being a centralized algorithm, it would introduce a significant communication overhead that would shorten the WSN lifetime, even though the SNs movement takes place only once. It should be also noted that the authors implicitly assume that the initial random deployment resulted in a connected network so that each SN is able to communicate with the base station.

The PSO-Voronoi deployment algorithm is further improved in[91]. The improved algorithm consists of two phases: phase I is the PSO-Voronoi deployment algorithm, called in this paper $WSNPSO_{vor}$, while phase II, called $WSNPSO_{per}$, aims at minimizing the collective energy consumed during the movement of mobile SNs to their optimized positions. $WSNPSO_{per}$ solves the following minimization problem:

$$min\ f_{energy} = dmax_{mov} \tag{2.35}$$

$$dmax_{mov} = max\{d_i\}\ \ , i = 1,2, \dots, N \tag{2.36}$$

where f_{energy} is the objective function of the optimization, $dmax_{mov}$ is the maximum moving distance required from any deployed SN after the conclusion of the first phase $WSNPSO_{vor}$, and N is the number of deployed SNs in the RoI. The operation of $WSNPSO_{per}$ starts by taking the following inputs: the initial positions of SN nodes after random deployment, their IDs and the final positions suggested by $WSNPSO_{vor}$. It then assigns each SN to one of the optimized final positions such that $dmax_{mov}$ is minimized using a simple form of PSO. Results showed a significant reduction of $dmax_{mov}$ in five different scenarios with varying number of deployed SNs, when using the two-phase algorithm versus using only phase I. Moreover, phase II does not noticeably increase the computation time or alter the total coverage achieved by phase I.

The study in [92] considers the problem of near-optimal deployment of multiple sink nodes in a pre-deployed static WSN, with the objective of minimizing the maximum worst case delay of SN-sink message communications. The authors assume that there are a finite number of candidate positions for the sink nodes. Hence, they formulated the optimization problem as follows:

$$min \quad max_{i\epsilon\{1,\dots,n\}}\{d_i\}, \tag{2.37}$$

where

$$d_i = f(\tau|\alpha,\beta) \tag{2.38}$$

$$\tau = g(s|p,R) \tag{2.39}$$

The parameter d_i is defined as the worst case delay of SN i. It is the objective function of the optimization problem. It is logically a function of the given topology of the WSN, τ, given a specific arrival and service patterns for each SN, represented by α and β respectively. The topology is in turn a function of the actual positions of the sink nodes, s given the vector, p which contains the SN positions in the RoI and the routing algorithm used, R. The authors propose using PSO with Local Search (PSO-LS) algorithm to find near-optimal solutions for the optimization problem expressed in (2.37) - (2.39). LS is used to escape the tendency of a conventional PSO to converge fast to a local optimal solution, but instead evolves to better solutions. The proposed algorithm is applied to several WSN topologies with different sizes, i.e. different numbers of SNs and sink nodes. The results of the PSO-LS algorithm were compared to the results obtained by applying a genetic-based algorithm with the same parameters. The comparison suggests a superior performance of PSO-LS in terms of convergence speed and quality of obtained solutions.

It is important to point out that the all the presented PSO-based deployment algorithms in this section so far have one major disadvantage in common, which is failing to account for the obstacles and/or preferential areas in a RoI. It was implicitly assumed in these algorithms that the terrain of the RoI is obstacle-free.

Unlike the algorithms reviewed so far in this section, which are based on PSO, the algorithms presented in [93] - [95] are based on ACO. In[93] , the authors present an ACO deployment algorithm called EasiDesign. The algorithm is designed to solve the *Minimum-Cost Connectivity-Guaranteed* SDP (MCCG-SDP). The authors assume that the RoI is modeled by a grid and that SNs can only be deployed on those grid points. To account for obstacles in the RoI, the authors exclude grid points associated with obstacles from the allowable set of grid-points at which SNs can be deployed. The algorithm constructs solutions to the problem, i.e. SNs layouts, by allowing the ants to transition from one grid point to another deploying SNs until full coverage, i.e. coverage of all grid-points in the RoI, is achieved. Hence, following the ACO mathematical framework discussed in Section 2.3.4.2, the directed graph of the algorithm $G(\boldsymbol{V},\boldsymbol{E})$ is constructed such that the set of vertices $\boldsymbol{V}$ is the allowable set of grid-points at which SNs can be deployed. As explained earlier in Section 2.3.4.2, ants' transitions between grid points are stochastic. Assuming the i^{th}ant is at a grid-point u on $G(\boldsymbol{V},\boldsymbol{E})$, the set of allowed successor grid-points is the set $\boldsymbol{N}_i^u \subset \boldsymbol{V}$ such that each grid-point $v \in \boldsymbol{N}_i^u$ is within an Euclidean distance equal to the SNs' communication range from grid-point u. Using this definition of $\boldsymbol{N}_i^u$ guarantees that the resulting solution/layout is connected to the sink node, assuming all ants start their tour at the sink node position. The probability that the i^{th} ant will transition from grid point u to point v is given by (2.40), where $\tau_{u,v}$ is the pheromone level between the two grid points, $\eta_{u,v}^i$ is a variable that is proportional to the coverage gain to deploying a SN in point v and α and β are constants.

$$p_{u,v}^i = \frac{[\tau_{u,v}]^\alpha [\eta_{u,v}^i]^\beta}{\sum_{m \in N_i^u} [\tau_{u,m}]^\alpha [\eta_{u,v}^i]^\beta} \ , v \in N_i^u \tag{2.40}$$

For the pheromone update procedure, the authors use a slightly version of the pheromone update rule expressed in (2.20) as follows:

$$\tau_{u,v}{}' = \tau_{u,v}(1-\rho) + \Delta_{u,v}^{best} \tag{2.41}$$

$$\Delta_{u,v}^{best} = \begin{cases} \frac{1}{L^{best}} \ , if\ u\ is\ in\ best\ solution \\ 0 \quad , \qquad\qquad otherwise \end{cases} \tag{2.42}$$

where L^{best} is the number of SNs in the best solution constructed in the last iteration of the algorithm. For performance evaluation, the authors applied EasiDesign to find an optimal SN layout for an environmental monitoring WSN. Their results indicated that EasiDesign algorithm can work efficiently in complex real-life applications. However, the presented results did not indicate the performance of EasiDesign in terms of speed of convergence. In addition, the authors did not compare the performance of EasiDesign in terms of optimality/quality of obtained solution to any other existing deployment algorithms.

In [94], the authors enhance the EasiDesign algorithm of [93]. They proposed an ACO deployment algorithm with three classes of ant transitions (ACO-TCAT) to solve the MCCG-SDP. The authors define the MCCG problem as finding the required minimum number of

SNs and their locations in a grid-based RoI to achieve full coverage of a given set of Points of Interest (PoIs) while guaranteeing that the resulting WSN is connected to a sink node in an arbitrary location in the RoI. The main difference between ACO-TCAT and EasiDesign is the definition of the the set of allowed successor grid-points $\boldsymbol{N}_i^u \subset \boldsymbol{V}$, for $i = 1,..,N_{ants}$ and $u \in \boldsymbol{V}$. ACO-TCAT uses three classes (i.e. three different definitions for $\boldsymbol{N}_i^u$) of ant transitions instead of one class in EasiDesign. This modification is introduced to enhance the search capability of ACO-TCAT, i.e. it enables the algorithm to converge faster. For all three classes of ant transitions, successive grid points in an ant's tour must be within a distance equal to the SNs communication range, i.e. neighboring grid-points. Similar to EasiDesign, this condition is used to guarantee that the obtained solutions are connected to the sink node. The first class of ant transitions restricts $\boldsymbol{N}_i^u$ to neighboring grid points with a coverage gain of one or more PoIs. If this class/set is empty, i.e. there are no neighboring grid-points that satisfy this condition, the second class is used. The second class of ant transitions restricts $\boldsymbol{N}_i^u$ to neighboring grid points that in turn have neighboring grid points with a coverage gain of one or more PoIs. If the second class is empty, then the third and final class is used. The third class of ant transitions is defined as neighboring grid points that have not been visited yet in the ant's tour. Presented results show that ACO-TCAT outperforms exiting greedy algorithms [29] in terms of the quality of obtained solutions. They also show that although ACO-TCAT does not outperform EasiDesign in terms of the quality of obtained solutions, it outperforms it in terms of the speed of convergence.

In [95], the authors address the blindness-of-connection problem which may occur when applying ACO-TCAT algorithm in [94] to solve an MCCG-SDP instance. This problem can occur during an ant's search when the ant is forced to apply the second or third transition class in the ACO-TCAT algorithm which can lead to the additions of redundant SNs to the final solution/deployment. To enhance the performance of the ACO-TCAT algorithm in terms of the quality of obtained solution (i.e. deployment cost), the authors in [95] propose a different set of ant transition rules and coin their proposed ACO algorithm the Node Deployment Strategy for Blindness Avoiding (NDSBA) algorithm. NDSBA is based on the same assumptions and problem formulation as ACO-TCAT algorithm in [94]. Both algorithms adopt the same transition probability rule expressed in (2.39) and the same pheromone update rule expressed in (2.40) and (2.41). The first class of ant transition in the ACO-TCAT algorithm is also the same as the basic ant transition class in NDSBA. However, NDSBA replaces the second and third ant transition classes in ACO-TCAT by three novel rules, namely: *greedy-migration, long-distance-jumping* and *short-distance-jumping*. An ant resorts to the greedy-migration transition rule when there are no neighboring grid points that have a coverage gain i.e., no Effective Candidate Points (ECPs). In this case, the ant jumps to a previously visited grid-point which has the highest number of ECPs within its communication range. In the case where greedy-migration is inapplicable, the grid-points visited by the ant thus far form a Local Connected Group (LCG) and the ant applies the long-distance-jumping rule. In this rule, the ant randomly chooses any unvisited grid-point which has a non-zero coverage gain i.e., an ECP. Due to this transition rule, a new LCG will arise which is not connected to the previous LCG(s). After all PoI are covered, the ant uses the *short-distance-jumping* transition rule to add grid points to its tour (i.e., add SNs to the deployment) such that the different constructed LCGs and the sink node form a connected graph following a simple greedy logic. After this step, the ant concludes its tour, pheromone levels are updated and a new iteration starts provided that convergence conditions are not yet satisfied. Results show that the NDSBA algorithm significantly outperforms ACO-TCAT algorithm in terms of the quality of the obtained solutions. The authors also prove that a single NDSBA algorithm iteration has the same computational complexity as that of an

ACO-TCAT algorithm iteration which is $O(n^2)$, where n is the number of grid points in the RoI.

The three ACO-based algorithms discussed above have two main advantages over the PSO-based algorithms reviewed in this section. The first advantage is the ease of factoring in the presence of obstacles in the RoI. This is due to the design of ACO which reduces the deployment problem to finding optimal path in a graph $G(\boldsymbol{V}, \boldsymbol{E})$ which is very compatible with the properties of SN deployment problems. The second advantage is that the obtained solutions, i.e. WSN layouts, from ACO-based algorithms are connected whereas this is not the case for all the PSO-based algorithms.

We conclude this section with Table 2.7, in which we compare between the deployment algorithms reviewed in terms of the type of SI method used (PSO or ACO), the characteristics of the targeted RoI, the sensing model adopted in coverage calculations and the algorithms' speed of convergence.

Table 2.7 Comparison among the SI algorithms for WSN deployment

Algorithm	Type Of SI Method	ROI	Sensing Model	Speed of Convergence
PSO-Grid in [89]	conventional PSO	w/o obstacles	deterministic	slow
PSO-Voronoi in [90], [91]	conventional PSO	w/o obstacles	deterministic	medium
PSO-LS in [92]	conventional PSO with LS	w/o obstacles	deterministic	fast
EasiDesign in [93]	ACO with one class of ant transitions	w/ obstacles and preferential areas	deterministic	medium
ACO-TCAT in [94]	ACO with three classes of ant transitions	w/ obstacles and preferential areas	deterministic	fast
NDSBA in [95]	ACO with four classes of ant transitions	w/ obstacles and preferential areas	deterministic	fast

2.5. Discussion and Experimental Evaluation

2.5.1. Discussion: Comparing the Four Approaches

Through the review and discussion presented in Section 2.4, it can be seen that the choice of a WSN deployment approach depends on several factors. These factors include the targeted RoI, whether a distributed or centralized approach is required, the degree of SN mobility (if any) in the WSN and whether the deployment has a single or multiple objectives. The complexity of the deployment approach is also a deciding factor in some WSN applications.

The targeted RoI involves whether the WSN is deployed in an obstacle-ridden or an obstacle-free RoI. It also involves whether the RoI is expected to be static or dynamic during the lifetime span of the WSN. Incorporating the presence of static obstacles, such as concrete walls, in the deployment problem is feasible in all four approaches. This is clear in the deployment algorithms which belong to the CG and APF approaches, in addition to ACO in the SI approach. As far as the GA and PSO are concerned, the presence of static obstacles can be handled by limiting the search space of the problem to a set of candidate deployment points or areas. It can also be achieved by introducing further constraints on the deployment

problem, or designing the fitness function such that it penalizes solutions with deployment positions on/near obstacle locations. The same argument is applicable to preferential areas.

On the other hand, the deployment of WSNs in highly dynamic and/or hostile RoIs requires the use of a deployment approach that provides the network with a self-organizing feature. APF-based algorithms, especially when implemented in a distributed fashion, have the ability to provide mobile SNs in MWSNs with real-time navigation in such environments without the need for a global localization method, albeit with a high cost. The high cost is partially due to the extended sensing capabilities required in the mobile SNs to detect their surroundings (e.g. a laser range-finder), and partially due to the repetitive movement of SNs which shortens the WSN lifetime. This can be overcome by deploying redundant SNs in the RoI which can be put to sleep and activated as needed. It should be noted that the performance of the distributed APF-based algorithms, in terms of energy consumption, are highly sensitive to the equilibrium conditions used to restrict unnecessary motion. Distributed CG-based deployment algorithms applied to MWSNs can also handle some changes in the RoI, particularly changes in the area coverage due to SNs dying from energy depletion and/or external attacks. As discussed earlier in Section 2.4.3, the performance of APF-based algorithms, in terms of area coverage and energy consumption, can be greatly enhanced by combining the conventional approach with CG (e.g. the use of Voronoi cells in local coverage calculation).

Since both GA and SI approaches are heuristic global optimization methods, they can only be implemented in a centralized fashion in WMSNs. This is not the case for APF and CG approaches, which have the option of being implemented in a distributed or centralized manner. This greatly limits the ability of GA and SI approaches to deal with dynamic RoIs.

Although all four approaches can be used in deploying mobile SNs, the use of either GA or SI approaches requires a powerful sink node or base station, since they can only be deployed in a centralized fashion. The sink would run the algorithms, based on a global knowledge of the SN locations and other parameters, and then communicates the new positions to the SNs to perform a one-time movement. This makes these two approaches unsuitable for applications targeting harsh or hostile RoIs, where the survival of a sink node is not guaranteed. For such applications, using a distributed CG or APF approach is more suitable.

As far as design objectives are concerned, GA and SI approaches are better suited for deploying WSNs with multiple design objectives than CG and APF approaches. This is due to the fact that multiple objectives can be easily factored in the fitness function used in a GA or SI algorithm. It should be pointed out that APF-based deployment algorithms can theoretically deal with multiple design objectives. This can be achieved by introducing more virtual forces to the algorithm to represent objectives besides optimizing the RoI coverage, for example maximizing the WSN throughput. However, adjusting the weights of these added virtual forces is a difficult process since they can only be estimated thorough trial and error.

In terms of the complexity, it would only be subjective to hold a general comparison between the algorithms which belong to the four different approaches. Although calculating the complexity of an APF or CG-based algorithm is a rather straightforward task as it is primarily a function of the number of deployed SNs, this is not the case for Genetic and SI algorithms. This is because the complexity of the latter two algorithms is a function of many variables. For GAs, it is a function of the size of the population, fitness function calculations and implementation of the genetic operators. Similarly, the complexity of a SI algorithm is a function of the number of particles in the swarm/ants in the ant colony, fitness function calculations and particles velocity/ants transitions calculations used for the swarm's evolution. Hence, a comparison can practically be held only among specific algorithms which follow the different approaches applied to the same deployment problem. To the best of our

knowledge, a comparative study of this description has not been published yet and it is one of our ongoing research studies. Table 2.8 summarizes the comparison among the different approaches. In this table, we compare the four approaches in our classification in terms of adaptability to changes in RoI, whether their implementation is centralized or distributed, suitability for deploying static and mobile WSNs, their ability to incorporate multiple network design objectives and the factors affecting their complexity.

Table 2.8 Comparison among the four mathematical approaches for planned WSN deployment

Mathematical Approach	Adaptability to RoI Changes	Centralized /Distributed	Suitability for Mobile/Static WSNs	Applicability to Multiple Objectives	Factors Affecting Complexity
GA	no	centralized	static; one-time movement	yes	no. of deployed SNs genetic operators fitness evaluation
CG	yes but limited	both	both	no	no. of deployed SNs
APF	yes	both	both	yes but difficult	no. of deployed SNs
SI	no	centralized	static; one-time movement	yes	no. of particles/ants swarm's velocity/ant's transitions calculations fitness evaluation

2.5.2. Experimental Evaluation

In this section, we conduct a performance evaluation study of four of the existing SN deployment algorithms that are designed to solve the MCC-SDP, namely the MAX-AVG-COV GH in [29], the *i*FLGA in [73], the *i*VLGA proposed in [75] and the EasiDesign ACO algorithm in [93]. The MCC-SDP is chosen since it is the most studied deployment cost minimization SDP in the literature. It is defined in Section 2.4.1 and mathematically formulated by (2.22) – (2.24). The *i*FLGA, *i*VLGA and EasiDesign algorithm represent two of the mathematical approaches in the presented classification in this chapter: GAs and SI. These two approaches are the most suitable of the four approaches for the planned deployment of *static* WSNs, where the SDP is modeled as a constrained optimization problem. The GH MAX-AVG-COV is used to benchmark the performance of these algorithms. These specific algorithms are selected based on the adaptability of their design to solve most versions of the MCC-SDP of different required coverage types (e.g. area coverage, point coverage or barrier coverage) and SN coverage models (e.g. binary disk model, FoV…etc.). The performance of the four algorithms is evaluated in terms of three metrics: quality of the obtained solutions (i.e. deployment cost), computational cost and speed of convergence. The results are then statistically analyzed and a comparison is conducted among the four algorithms.

2.5.2.1. Experimental Set-up

To evaluate the performance of the four algorithms, we implement and apply each to six different scales of the MCC-SDP, where the RoI is modeled by a square grid ($M = N$) with 10, 15, 20, 25, 30 and 35 gird points in each dimension respectively. At all tested problem

scales, we assume that grid points are 5 meters apart and that the SNs have a coverage range r_s of 15 meters. The SNs communication range r_c is set to 30 meters in EasiDesign. This assumption results in a ratio $r_c : r_s$ of 2. Consequently, the obtained solutions (i.e. SN deployments) from all four algorithms will provide both comprehensive coverage and connectivity among SNs. The algorithms were developed on MATLAB R2010b version 7.11.0.584. To account for the stochastic nature of the algorithms, each is executed for 10 trials on an Intel Xeon processor, CPU E5620, 2.4 GHz and 12 GB RAM.

We use the same parameter settings specified for iFLGA, iVLGA and EasiDesign in [73], [75]and [93] respectively. To ensure a fair comparison in terms of the computational cost and speed of convergence, we use the same termination conditions for all three metaheuristic algorithms. We use two termination conditions in our experiments. The first one is the algorithm reaching a maximum number of 500 iterations. The second condition is the convergence of the algorithm, signaled by the stagnation of the maximum fitness through 100 iterations. Performance is evaluated in terms of three metrics:

- *Quality of obtained solutions*, which is measured by the number of SNs in the minimum-cost deployments obtained by the four algorithms. For the metaheuristic algorithms iFLGA, iVLGA and EasiDesign, this number is equivalent to highest fitness achieved at the algorithm termination.
- *Computational cost*, which is measured using the CPU run-time required for an algorithm to terminate or converge.
- *Speed of convergence*, which pertains only to the three metaheuristic algorithms. This metric is measured by the number of iterations the algorithm executes before converging (if convergence occurs). This metric does not apply to MAX-AVG-COV since it terminates once a solution to the problem is found.

We also investigate how these three performance metrics change with the increase of the problem scale, i.e. algorithm scalability.

2.5.2.2. Results and Discussion

We now present and discuss the obtained results according to the aforementioned performance metrics.

Quality of obtained solutions: Table 2.9 summarizes the results in terms of the quality of the obtained solutions, showing the lowest (best), highest (worst) and the average number of SNs in the solutions obtained by the four algorithms. For each problem scale, the lowest obtained average (i.e. lowest average deployment cost) is written in bold font. The three metaheuristic algorithms outperform the MAX-AVG-COV GH at all tested scales of the problem. This outcome is expected since the evolutionary nature of the metaheuristic algorithms makes them generally more capable than GHs of finding higher quality solutions to difficult optimization problems. The performance gap between the three metaheuristic algorithms and MAX-AVG-COV increases progressively as the problem. The algorithm EasiDesign and iFLGA demonstrate the best performance, outperforming iVLGA with a small margin at the largest three scales of the problem.

Based on the average solutions quality, both $EasiDesign$ and iFLGA show a similar performance. However, EasiDesign exhibits a higher variability in the quality of its solutions. To provide a more statistically accurate comparison, a set of pair-wise t −tests is performed to confirm the initial observations drawn from the results in Table 2.9.

Table 2.9 Comparison among the four algorithms in terms of quality of obtained solutions

Grid Size (MxN)	MAX-AVG-COV			*i*VLGA			EasiDesign			*i*FLGA		
	Best	**Avg.**	**Worst**	**Best**	**Avg.**	**Worst**	**Best**	**Avg.**	**Worst**	**Best**	**Avg.**	**Worst**
(10x10)	6	6	6	4	4.8	5	4	5.1	6	4	**4.6**	5
(15x15)	14	14.2	16	11	12.7	14	11	12.5	14	10	**11.6**	13
(20x20)	26	26.9	28	22	24	26	21	**23.2**	25	22	**23.2**	24
(25x25)	39	40.2	41	35	37.1	38	32	**35.3**	38	34	35.6	37
(30x30)	57	58	61	51	54.2	57	49	**51.9**	54	50	52.2	54
(35x35)	76	78.5	80	70	74.5	78	68	72.9	77	68	**72**	74

Table 2.10 Results of the pairwise t −tests

Grid Size (MxN)	MAX-AVG-COV Vs. *i*VLGA[1]	*i*VLGA Vs. *i*FLGA[2]	*i*FLGA Vs. EasiDesign[3]	Conclusion
(10x10)	4.27×10^{-6}	1.80×10^{-1}	9.80×10^{-2}	MAX-AVG-COV< *i*VLGA≈ *i*FLGA≈EasiDesign
(15x15)	2.40×10^{-4}	1.82×10^{-2}	5.04×10^{-2}	MAX-AVG-COV< *i*VLGA≈ *i*FLGA≈EasiDesign
(20x20)	1.57×10^{-4}	1.10×10^{-1}	10×10^{-1}	MAX-AVG-COV< *i*VLGA≈ *i*FLGA≈EasiDesign
(25x25)	1.26×10^{-6}	2.34×10^{-3}	6.35×10^{-1}	MAX-AVG-COV< *i*VLGA< *i*FLGA≈EasiDesign
(30x30)	4.72×10^{-5}	7.21×10^{-3}	6.52×10^{-1}	MAX-AVG-COV< *i*VLGA< *i*FLGA≈EasiDesign
(35x35)	1.02×10^{-4}	7.83×10^{-3}	4.03×10^{-1}	MAX-AVG-COV< *i*VLGA< *i*FLGA≈EasiDesign

All t −tests are carried out at a 95 percent confidence interval, i.e. α = 0.05
"A< B" means B performs better than A, while "A≈B" means A and B perform similarly
1 p −values of the one-tailed t −test of the alternative hypodissertation that the average of *i*VLGA is greater than that of MAX-AVG-COV
2 p −values of the one-tailed t −test of the alternative hypodissertation that the average of *i*FLGA is greater than that of *i*VLGA
3 p −values of the two-tailed t-test of the alternative hypodissertation that the averages of *i*FLGA and EasiDesign are unequal

Table 2.10 shows the resulting p −values of the performed t −tests and the corresponding conclusion for each problem scale. Values less than the specified statistical significance α=0.05 indicates that the null hypodissertation of equal averages is rejected and that the alternative hypodissertation of the test is true with a 95% confidence level. Conclusions in Table 2.10 coincide with our initial observations: in terms of quality, the ascending order of performance is $\mathrm{MAX-AVG-COV} < i\mathrm{VLGA} \approx i\mathrm{FLGA} \approx \mathrm{EasiDesign}$ for the three smallest problem scales and MAX-AVG-COV< *i*VLGA< *i*FLGA≈ EasiDesign for the three largest scales, where "A< B" means B performs better than A, while "A≈B" means A and B perform almost similarly. The deterioration of the *i*VLGA performance in terms of quality at larger problem scales can be attributed to the variable-length chromosome encoding scheme. As the scale of the problem increases, i.e. as the number of grid points increases, the probability that one or more genes belonging to the global optimum solution are not represented in the initial population increases as well. Since the crossover operator simply recombines existing chromosomes, the mutation operator is the only source of

diversification in the algorithm. At larger problem, it may not be sufficient to successfully guide the algorithm's search to higher quality solutions.

Computational Cost and Speed of Convergence: These two performance metrics are closely correlated and together they determine the computational efficiency of a stochastic optimization algorithm. Fig. 2.7 illustrates the performance of the four algorithms in terms of the computational cost measured by the average CPU run-time in seconds. Fig. 2.8 illustrates the performance of the three metaheuristic algorithms in terms of their speed of convergence, measured by the number of iterations executed by the algorithm before termination (either by convergence or reaching the maximum number of iterations). In Fig. 2.8, each result point consists of the average, upper and lower limits of the 95% confidence interval of the corresponding set of runs.

In terms of computational cost, the ascending order of performance is EasiDesign $<$ iFLGA$<$ iVLGA$<$ MAX-AVG-COV. As a GH, the low computational cost of MAX-AVG-COV is expected and attributed to its simple design. Compared to iFLGA and EasiDesign, iVLGA has a much lower computational cost. This can be attributed to the variable-length chromosome encoding scheme, which significantly decreases memory assignment time during the iVLGA execution, reducing its computational cost. This is not the case for both iFLGA and EasiDesign, where the fixed chromosome length and the average number of ants' transitions are directly proportional to the problem scale. Theoretically, the performance gap between the iVLGA and both iFLGA and EasiDesign in terms of computational cost should increase with the increase of the problem scale. However, it is interesting to observe that in practice this is not valid for iFLGA as the difference between it and iVLGA decreases steadily with the increase of the scale. It can also be observed that iFLGA and EasiDesign have a comparable computational cost for the smallest problem scale, but as the scale grows, iFLGA starts exhibiting a steadily increasing advantage over $EasiDesign$. These two observations can be explained by Fig. 2.8. In the figure, it can be seen that iFLGA shows the best performance in terms of the speed of convergence with a large margin over both iVLGA and EasiDesign. For the majority of the runs at all tested problem scales, iFLGA converges within 200 iterations.

On the other hand, the ability of iVLGA and EasiDesign to converge quickly deteriorates as the problem scale grows. This explains the narrowing in the performance gap in terms of the computational cost between iVLGA and iFLGA: although the average time to execute iteration in iVLGA is always lower than in iFLGA, the slow convergence of the iVLGA dampens its advantage over iFLGA. The deterioration in convergence speed of the iVLGA at larger problem scales can be attributed again to the variable-length chromosome encoding scheme: the limited level of diversification introduced by the genetic operators in iVLGA slows down its convergence. For EasiDesign, the slow convergence (at the three smallest problem scales) and the lack of it (at the three largest problem scales) can be due to the definition of the set of allowed successor grid points $\boldsymbol{N}_i^k$. The adopted definition of this set limits the successor grid points and hence creates a situation where a grid point that offers no coverage gain can be included to an ants' tour (i.e. solution). This in turn leads to redundant transitions which slows down the convergence of the algorithm or prevents it altogether.

2.6. Chapter Summary

In this chapter, we surveyed and classified the planned WSN deployment algorithms which have been presented in the literature according to their mathematical approach. Four distinct approaches were proposed for this classification, namely GAs, CG, APFs and SI. We

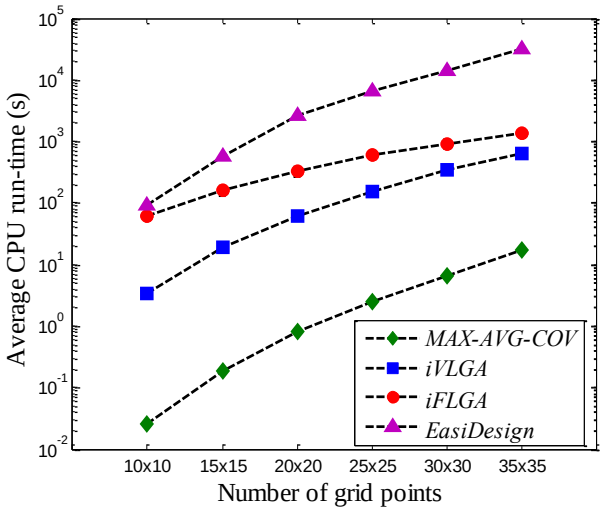


Fig. 2.7 Comparison among the four algorithms under consideration in terms of computational cost measured by the average CPU run-time.

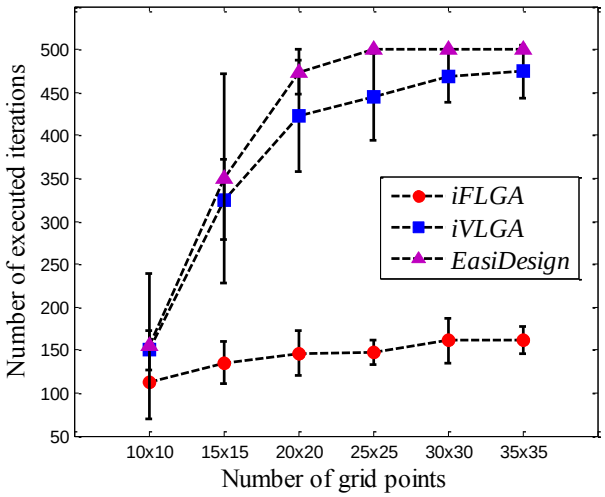


Fig. 2.8 Comparison among the three metaheuristic algorithms under consideration in terms of convergence speed measured by the number of executed iterations.

discussed some of the fundamental design factors of WSNs, namely the sensing model, mobility of SNs, WSN coverage and network connectivity. We presented a brief account on the background and mathematical foundation of each of the four approaches. An extensive review of the deployment algorithms which belong to each approach was presented. In this review, we presented comparisons between the different deployment algorithms based on each approach. We then discussed and compared the four approaches in terms of different WSN design factors, thus highlighting the strengths and limitations of each approach. One of the most important conclusions drawn from the conducted survey is that GAs and SI algorithms, specifically ACO algorithms, are best suited for deploying static WSNs with single or multiple design objectives. Finally, we presented and discussed a performance evaluation study of four of the existing SN deployment algorithms.

Chapter 3

Reliability Assessment of Wireless Sensor Network Deployments

3.1. Introduction

Evaluating the reliability of WSNs is of great importance especially for WSNs designed for mission-critical applications. For such applications, the failure of the WSN to perform its function(s) can have catastrophic effects [6]. In this chapter, we review and discuss existing studies on the reliability and fault-tolerance of WSNs. Based on the presented review, we propose a novel WSN reliability metric in terms of the coverage and connectivity of the network. This is done under the practical assumption that that an SN functions as a three-mode device (*on*, *off* and *relay* modes) as opposed to the conventional two-mode SN model (*on* and *off* modes) in the existing work in the literature.

The reliability of any multi-component system is formally defined as the "probability that a system will perform satisfactorily during its mission time when used under the stated conditions" [19]. The method by which the reliability of a specific system is evaluated varies according to the type(s) of components of which the system is composed, the configuration of the system in terms of how these components are connected to each other and the state(s) at which the system is defined to have failed. Ultimately, the reliability of the system is a function of the reliability measures of its components. Therefore, evaluating the reliability of the system as a whole is a probability-modeling problem. In this context, a WSN can be viewed as a multi-component system in which the components are the sensor nodes (SNs) and the sink node(s). The mission time for a WSN can either be its intended lifetime or the time interval between scheduled maintenance operations. Hence, the WSN mission time is application-dependent and can vary greatly ranging from a few days to a few years. The configuration of the WSN is determined by the way the SNs are deployed in the targeted RoI and the resulting wireless connectivity among them.

In order to identify the states at which a given WSN deployment fails, the functionality of a WSN must first be defined. The functionality of a WSN can be divided into two major elements. The first element is the sensing functionality, which is the ability of a WSN to detect all the targets or phenomena that occur inside the boundaries of the RoI during its mission time. Hence, for a WSN to be functional in terms of sensing it must provide full coverage for the RoI area (in case of area coverage) or all the targeted locations in the RoI (in case of point coverage) during its mission time. The second element of the WSN functionality is the connectivity functionality, which is the ability of the WSN to deliver sensed data from its sources (i.e. SNs) to the designated destination (i.e. sink node(s)) during its mission time. Hence, for a WSN to be functional in terms of connectivity, any target or a phenomenon detected by one or more SNs in a WSN has to be recognized at the sink node(s) through multi-hop wireless communication throughout the WSN mission time. Based on this definition of the WSN functionality, a WSN is said to have failed if either of its sensing or connectivity functionality elements fails [96].

There are several issues that affect the reliability of a WSN. These issues can generally be classified into SN related and non-SN related issues. The SN related issues are factors pertinent to the functionality of the deployed SNs, namely, SN power failure, hardware

failures and software failures. The effect of these issues on the functionality of the network during its mission time (i.e. on the reliability of the network) can be predicted [97] as will be discussed later. These issues can be summarized as follows:

- SN Power Failure: the majority of the industrial and commercial SNs currently available in the market are battery-powered. Current advances in the fabrication of batteries have recently introduced highly durable batteries for SNs that can last for years (e.g. lithium thionyl chloride batteries [98]) under certain conditions. Although, theoretically, these batteries can sustain the operation of the SNs for long periods of time, premature battery failures can still occur in practice. This can be attributed to a myriad of reasons such as the deployment of the SNs in harsh environmental conditions (e.g. extreme temperatures or rain), incorrect handling or random failure caused by defective hardware [99].
- SN Hardware Failures: SNs are subject to random hardware failures. This is attributed to two main reasons. The first one is that most commercial SNs are cost-sensitive, meaning that they are not always built of the highest quality components. The second reason is that SNs are often subjected to harsh environmental conditions which can affect the normal operation of its components [20].
- SN Software Failures: SNs are prone to random software failures which can render them inactive, i.e. unable to sense or communicate.

On the other hand, non-SN related issues are factors that are external to the deployed SNs such as wireless link failures (due to fading and external interference) and excessive packet collisions (i.e. internal interference in WSNs adopting contention-based medium access control). The effect of these issues on the overall network reliability is in general difficult to predict [100]. However, several measures can be adopted to mitigate their detrimental effect on the network reliability. Examples of such measures include acknowledgements and retransmissions.

3.2. Related Work on Reliability and Fault Tolerance of WSNs

Several studies have addressed the issue of evaluating or estimating the reliability of WSNs. In this section, we review the most significant of these studies and discuss their scope and limitations.

In the studies presented in [101] - [103], the authors use a reliability metric to measure the reliability of SN systems used in monitoring linear processes in chemical plants to cost-optimize the SN system layout. The proposed metric is formulated in terms of the failure probabilities of the SNs and depends on the method by which the different variables of a chemical process are measured and how they contribute to the monitoring of the chemical process. Hence, the metric is specifically tailored for this type of monitoring application and cannot be extended to other applications. Moreover, the authors consider SN systems and not networks, meaning that wireless communication between SNs is overlooked and not factored in the proposed reliability metric.

The studies in [104] and [105] address the problem of evaluating the reliability of WSNs characterized by cluster-based deployments subject to random SN failures. In both studies, the authors assume that the SN clusters are non-overlapping and that each cluster has a designated cluster head which acts as a relay between the SNs in the cluster and the sink node. In [104], the authors define the reliability of a cluster as the probability of successful message delivery between the sink node and the cluster head. The authors in [105] define the reliability of the WSN as the probability that the geographical area of each cluster in the WSN is fully covered by its SNs and that the cluster head has at least one functional direct or multi-hop wireless path to the sink node. Based on this definition, they derive an expression

for the reliability of each individual cluster and use a Monte Carlo (MC) simulation approach to estimate it. The main limitation of the studies in [104] and [105] is that the reliability of the WSN as a whole in terms of the reliability of its constituent clusters is not evaluated. In addition, the proposed definitions of reliability cannot be extended to WSNs with different deployment configurations such as flat deployments which are non-hierarchical.

On the other hand, the studies in [106] - [110] address the reliability of SN systems or WSNs of non-hierarchical deployment configurations. In [106], the authors address the problem of evaluating the reliability of WSNs designed for industrial inventory management. They assume that, for the purposes of this specific application, the data collected by each SN are stored redundantly on several other SNs to account for random SN failures. Accordingly, the WSN is deemed functional as long as there is a sufficient number of functional SNs that are both connected to each other and to the sink node. Based on this definition of network functionality and the assumption that the WSN deployment is homogeneous, the reliability evaluation problem is reduced to the famous K-out-of-N reliability problem [111]. The authors also present a MC simulation approach similar to that proposed in [105] to estimate the reliability of the WSN at hand. However, the reliability evaluation and estimation approaches proposed in [106] are based on a very restrictive definition of network functionality. Consequently, they cannot be applied to other WSN applications (e.g. surveillance and monitoring applications) where the functionality of the network is dependent not only on the number of SNs connected to the sink node but also on the network coverage. Also, the proposed approaches do not support network heterogeneity which is a major limitation since real-world deployments are often heterogeneous.

The authors in [107] propose a reliability metric for SN systems designed for surveillance purposes subject to random SN failures. They assume an arbitrary deployment configuration where SNs can monitor multiple target locations in the RoI and that each target location can be monitored by multiple SNs. They also assume that the surveillance SN system can be heterogeneous. The reliability of the system is defined as the probability that all target locations are monitored by at least one SN. The authors use a combinatorial approach to evaluate the proposed reliability measure. The main limitation of the proposed metric is that system functionality is assumed to be in terms of the degree of target locations coverage only. Connectivity between SNs to form a wireless network is not considered. Hence, the proposed approach cannot be applied to evaluate the reliability of a WSN deployment.

In [108], the authors propose a model for evaluating the reliability of a WSN subject to two types of failure events, namely, SN failures due to battery depletion and link failures. Their proposed approach depends on dividing the targeted RoI into disjoint areas or target regions. For each region, a reliability model is constructed using a Reliability Block Diagram (RBD) [19], which depends on the number of SNs monitoring the target region, their relative location from the sink node and the routing protocol used in the network. There are two drawbacks of the proposed reliability modeling proposed in [108]. The first drawback is that the model does not provide a method by which the reliability of the entire WSN deployment can be evaluated in terms of the reliability of its regions. The second drawback is that the reliability modeling is carried out under the assumption that the probabilities of link failures are known and are constant throughout the lifetime of the WSN. This assumption is unrealistic for wireless links since link quality is affected by numerous factors such as multi-path effects, shadowing (due to static and mobile obstacles) and interference. The effect of these factors on link quality varies significantly and rapidly in time and space [100] and hence, unlike SN related factors, cannot be reduced, contrary to hardware components, to a constant probability of failure throughout WSN mission time.

The study in [109] propose a method for evaluating the reliability of WSNs designed for industrial IoT applications based on the automatic generation of Fault Trees (FTs). The proposed method requires the network failure conditions as inputs to enable the generation of the corresponding network FT and compute the network reliability. A network failure condition is defined as a combination of SNs which if fail will lead to the failure of the WSN in terms of network coverage only and not connectivity to the sink. To address the connectivity part of the network functionality, the authors propose a depth-first search algorithm that finds all the paths between SNs belonging to the network failure conditions and the sink node. The study in [109] is extended in [110] by assuming that the WSN is also subject to permanent wireless link failures in addition to SN failures under the same assumptions adopted in [105]. However, the reliability metric proposed in [109] and [110] can not be used in the context of stochastic optimization and hence to calculate the reliability in the problem at hand since developing/constructing the FT and using it to calculate the paths set and the reliability of the WSN is a very time-consuming process.

3.3. Motivation for a New Reliability Metric

Based on the above discussion, existing studies provide reliability evaluation or estimation for WSNs under restrictive conditions that pertain to specific applications, network functionality definition and/or deployment configurations. More importantly, they all assume that SNs have only two modes of operation, either *on* or *off*. If an SN is *on*, it is assumed to be functional in terms of both sensing its surrounding environment and communicating wirelessly with its neighbors. If it is *off*, then the SN has failed permanently due to one or more of the SN related reliability issues outlined in the introduction of this chapter. This representation is not accurate since most commercial SNs are composed of multiple independent chips that carry out different functions, with each having its own probability of failure during the network's mission time.

A more accurate model considers the SN as a multi-component system [113]. Based on this model, an SN has three modes of operation. These modes of operation are *on*, *relay* and *off*. The definitions of the *on* and *off* modes are the same as discussed above, while the *relay* mode occurs when the SN is unable to perform its sensory function but it still able to communicate wirelessly with its neighbors. This mode of operation occurs when the SN's sensor(s) hardware fails while its transceiver, processor and battery are in working condition. Adopting this SN model provides a more accurate evaluation of WSN reliability, assuming that the network functionality is adequately defined in terms of both network coverage and connectivity.

3.4. Fundamental Reliability Concepts

In this section, we discuss some of the fundamental definitions and concepts related to the evaluation of multi-component systems' reliability which we will be using throughout this chapter.

3.4.1. Component Reliability Function and Component Reliability

The main objective of reliability modeling is to express the reliability of a given system in terms of the reliability measures of its constituent components. There are two main reliability measures for any device or component. The first measure is the reliability function, $R_c(t)$, which is used to estimate the probability that the device or component will continue to function beyond a time duration of length t [19]. The second reliability measure is based on the fact that, for most practical purposes, a device or component is only required

to function during the specified mission time T_m of the system it belongs to. In this case, the reliability function $R_c(t)$ can be substituted by the reliability of the device. The reliability of a device, R_c, is simply defined as the probability that the device will continue to function throughout the mission time of the system. Accordingly, the probability of failure of the device during T_m is equal to $1 - R_c(T_m) = 1 - R_c$ [19].

For example, Fig. 3.1 shows an exponential reliability function, which is one of the simplest functions used in modeling the reliability of electronic components. The exponential reliability function is $R_{c_e}(t)$ is given by the following equation:

$$R_{c_e}(t) = e^{-\alpha t} \tag{3.1}$$

where α is the estimated failure rate of the component per unit of measurement (e.g. hour, year, cycle…etc.) and is equal to the reciprocal of its Mean Time To Fail (MTTF). From the reliability curves in Figure 1, we can estimate the reliability at $t = 5000$ hours for $\alpha = 1/4000$ (i.e. for MTTF = 4000) to be 0.287. This in turn means that there is a $1 - 0.287 = 0.713$ chance that the component will fail during this time interval, i.e. the probability of failure during this time interval is 0.713.

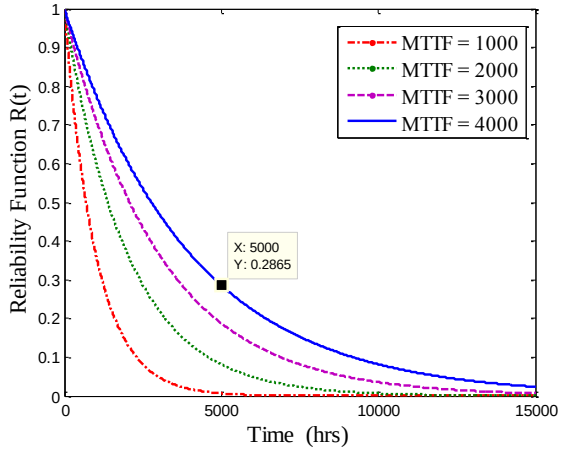


Fig. 3.1 Exponential reliability function plot for different values of MTTF (1000, 2000, 3000 and 4000) in hours.

Although the exponential reliability function is commonly used in reliability engineering due to its simplicity, it usually leads to inaccurate estimations of the probabilities of failures. This is because this type of function is based on the assumption that the component has a constant failure rate, which means that its performance does not degrade with time. To obtain a more accurate model for the reliability function of a given electronic device, reliability engineers carry out rigorous reliability testing techniques and/or gather empirical data on the device in service [19]. For example, qualitative and quantitative accelerated reliability testing is used to identify probable hardware failures of SNs and estimate the probability of their occurrence [97].

3.4.2. Combinatorial Approach to System Reliability Evaluation

Combinatorics is a proven useful tool in evaluating and estimating the reliability of complex systems and networks [114],[115]. The fundamental premise of the combinatorial approach to reliability evaluation is that the reliability of any system can be computed by means of evaluating the system's structure function for every possible state of the system. To explain this concept, consider a system $\boldsymbol{S}$ which consists of n components, i.e. $\boldsymbol{S} =$

$\{1,2,\dots,n\}$. Each component can only have two distinct states; it can either be functional or *on* or it can fail or be *off*. Let the binary variable π_i be the state indicator of component i as follows:

$$\pi_i = \begin{cases} 1, & if\ component\ i\ is\ on \\ 0, & if\ component\ i\ is\ off \end{cases} \tag{3.2}$$

A state $\boldsymbol{\pi}$ of the system $\boldsymbol{S}$ is a description of the states of all its components, hence $\boldsymbol{\pi} = \{\pi_i\}$ for $i = 1, \dots, n$. Let $\boldsymbol{\Pi}$ be the set of all possible states of $\boldsymbol{S}$. The structure function of $\boldsymbol{S}$, denoted $f(\boldsymbol{\pi})$, is a binary function that indicates whether the system is working under a given state according to the following equation:

$$f(\boldsymbol{\pi}) = \begin{cases} 1, & \boldsymbol{S}\ is\ functional \\ 0, & \boldsymbol{S}\ has\ failed \end{cases} \tag{3.3}$$

Based on the above definitions, the reliability of $\boldsymbol{S}$, denoted by $R(\boldsymbol{S})$, can be calculated using the following equation:

$$R(\boldsymbol{S}) = Prob(f(\boldsymbol{\pi}) = 1) = \sum_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} f(\boldsymbol{\pi}).Prob(\boldsymbol{\pi}) \tag{3.4}$$

To calculate $R(\boldsymbol{S})$ using (3.4), the conditions necessary for $\boldsymbol{S}$ to be functional must be defined and the probability of any system state must be evaluated in terms of the reliabilities (or probabilities of failure) of system's components, assuming that the system has a specified mission time T_m. Theoretically $f(\boldsymbol{\pi})$ must be evaluated for all the possible system states $\boldsymbol{\pi} \in \boldsymbol{\Pi}$ to calculate $R(\boldsymbol{S})$ using this approach. However, following this extensive method in reliability calculation poses a computational problem for systems of a practical scale. For example, a system composed of 30 components which fail independently has 2^{30} states. Therefore, a tremendous amount of time is required to calculate $R(\boldsymbol{S})$ which grows exponentially with the number of components in the system. This computational problem is mitigated by the use of more efficient methods (e.g. RBD, FT and search algorithms) that attempt to find all the system's path-sets or cut-sets [115].

To define a system's path and cut, let $\boldsymbol{S}_1(\boldsymbol{\pi})$ be the set of functioning components, i.e. components in the *on* state, in $\boldsymbol{S}$ for a given system state $\boldsymbol{\pi}$ and $\boldsymbol{S}_0(\boldsymbol{\pi})$ be the set of failed components, i.e. components in the *off* state. $\boldsymbol{S}_1(\boldsymbol{\pi})$ and $\boldsymbol{S}_0(\boldsymbol{\pi})$ can be expressed by the following equations:

$$\boldsymbol{S}_1(\boldsymbol{\pi}) \equiv \{i \mid \pi_i = 1,\ i \in \boldsymbol{S}\} \tag{3.5}$$

$$\boldsymbol{S}_0(\boldsymbol{\pi}) \equiv \{i \mid \pi_i = 0,\ i \in \boldsymbol{S}\}, \tag{3.6}$$

where $\boldsymbol{S}_1(\boldsymbol{\pi}) \cup \boldsymbol{S}_0(\boldsymbol{\pi}) = \boldsymbol{S}$. A state $\boldsymbol{\pi}$ of the system $\boldsymbol{S}$ is called a *path* if the system is functional at that state, i.e. $f(\boldsymbol{\pi}) = 1$. In that case, the corresponding *path set* is the set $\boldsymbol{S}_1(\boldsymbol{\pi})$, which is defined as the set of components whose simultaneous functional state guarantees that the overall system is functional. On the other hand, a state $\boldsymbol{\pi}$ of the system $\boldsymbol{S}$ is called a cut if the system fails at that state, i.e. $f(\boldsymbol{\pi}) = 0$. In this case, the corresponding cut set is the set $\boldsymbol{S}_0(\boldsymbol{\pi})$, which is defined as the set of components whose simultaneous failure results in the failure of the overall system. If all the path sets or alternatively all the cut sets of a system $\boldsymbol{S}$ are known, we can rewrite (3.4) as follows:

$$R(\boldsymbol{S}) = \sum_{\boldsymbol{\pi}\in\boldsymbol{\Pi}_1} Prob(\boldsymbol{\pi}) = 1 - \sum_{\boldsymbol{\pi}\in\boldsymbol{\Pi}_0} Prob(\boldsymbol{\pi}), \tag{3.7}$$

where $\boldsymbol{\Pi}_1$ is the set of all the paths of $\boldsymbol{S}$ (i.e. the complete paths set of $\boldsymbol{S}$) and $\boldsymbol{\Pi}_0$ is the corresponding set containing all the cuts of $\boldsymbol{S}$ (i.e. the complete cuts set of $\boldsymbol{S}$) such that $\boldsymbol{\Pi}_1 \cup \boldsymbol{\Pi}_0 = \boldsymbol{\Pi}$. For example, a simple system of n components connected in series has only one path set which is equal to the system set $\boldsymbol{S} = \{1,2,\dots.,n\}$ and has $\sum_{k=1}^{n} C_k^n$ cut sets. Therefore, it is simpler to express its reliability as $R(\boldsymbol{S}_{series}) = Prob(\boldsymbol{\pi} = \{\pi_i = 1,\ \forall\, i = 1,\dots,n\}) = \prod_{i=1}^{n} R_i$, where R_i is the reliability of the i^{th} component during the system's mission time. On the other hand, a system of n components connected in parallel has only one cut set which is equal to $\boldsymbol{S}$ and has $\sum_{k=1}^{n} C_k^n$ path sets. Hence, the system's reliability can be expressed as $R(\boldsymbol{S}_{parallel}) = 1 - Prob(\boldsymbol{\pi} = \{\pi_i = 0, \forall\, i = 1,\dots,n\}) = 1 - \prod_{i=1}^{n}(1 - R_i)$.

3.5. WSN Reliability Metric

In this section, we use the combinatorial approach outlined in Section 3.4 to derive our proposed WSN reliability metric for an arbitrary WSN deployment configuration. We start by presenting the adopted model for the WSN. We then derive a WSN reliability metric based on the assumption that the constituent SNs are three-mode devices characterized by two failure probabilities, namely, the sensor failure probability and the transceiver sensor probability, as an intermediate step. We will refer to this SN model as the 3-mode, 2-par model. Finally, we derive the proposed WSN metric based on the assumption that the constituent SNs are three-mode devices characterized by four failure probabilities, namely sensor, transceiver, processor and battery failure probabilities. We will refer to this SN model as the 3-mode, 4-par model.

3.5.1. WSN Model and Functionality Definition

We assume that the targeted RoI of the WSN is a two-dimensional area in which there is a finite set of locations that require some form of monitoring (e.g. motion, image, video...etc.) using static SNs. These locations are called target points and are denoted by the set $\boldsymbol{T} = \{t_j\}$ for $j = 1,\dots,m$. To maintain generality, we do not assume that the target points conform to any regular pattern. Target points are monitored by the SNs in the WSN. We assume that the SNs used in the deployment of the WSN can be of different types (e.g. sound, image, etc.) and can have different coverage profiles (e.g. binary disk model, FoV model, etc.), i.e. the WSN can be heterogeneous in nature. Let the set of deployed SNs be denoted by $\boldsymbol{S} = \{s_i\}, i = 1,\dots,n$. We assume an arbitrary deployment configuration in which an SN can monitor multiple target points. We also assume that a target can be monitored by more than one SN. Therefore, in terms of coverage, the WSN can be modeled as a bipartite graph. Fig. 3.2 shows an example of a WSN consisting of 5 SNs ($n = 5$) monitoring 3 target points ($m = 3$) and the resulting bipartite graph representation of the network coverage, assuming SNs are characterized by a binary disk coverage model. All sensory data acquired by the SNs should be relayed to a sink node with an arbitrary fixed position in the RoI through wireless multi-hop communications. We assume that all deployed SNs have a fixed communication range, r_c. Hence, any two SNs deployed have a wireless communication link if the distance between them is less than or equal to r_c. Naturally, it is required that the WSN remains functional in terms of coverage and connectivity throughout its intended mission time T_m. To express this mathematically, we use the following definition:

Definition 1: A WSN is said to be functional in terms of both coverage and connectivity if both of the following two conditions are met:

1. Each target point t_j for $j = 1, \ldots, m$ is covered by at least one SN with an uncompromised sensing capability, i.e. an SN in the on state. Let the set $\boldsymbol{Y}_j$ be the set of SNs in the on state that monitor t_j. Then this condition can be expressed as, $\left| \boldsymbol{Y}_j \right| \neq 0, \forall\, j = 1, \ldots, m$ where $|.|$ denotes the size of a set.
2. Within each $\boldsymbol{Y}_j$, there is at least one SN that has at least one functional path to the sink node. This implies that SNs along that path, including the source SN, have uncompromised communication capabilities, i.e. in either the on or the relay state. Hence, the events detected at any t_j can be relayed back to the sink node. Let the set $\boldsymbol{Z}_j$ be the set of SNs which belong to $\boldsymbol{Y}_j$ that are connected to the sink node. Hence $\boldsymbol{Z}_j \subseteq \boldsymbol{Y}_j$. The condition can be expressed as $\left| \boldsymbol{Z}_j \right| \neq 0, \forall\, j = 1, \ldots, m$.

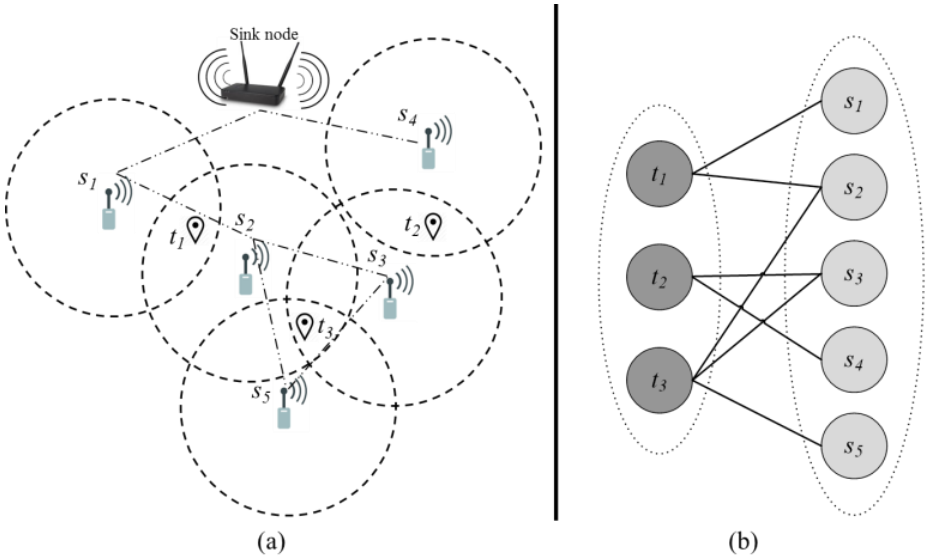


Fig. 3.2 (a) A simple WSN consisting of a sink node, 5 SNs ($n = 5$) and 3 target points ($m = 3$) ; (b) The coverage of the WSN is modeled as a bipartite graph in which the target point set $\{t_1, t_2, t_3\}$ and the SN set $\boldsymbol{S} = \{s_1, s_2, s_3, s_4, s_5\}$ are the two disjoint sets.

3.5.2. Reliability Metric Formulation for the 3-mode, 2-par SN Model

We now use the above definition of WSN functionality conditions in deriving a reliability metric for arbitrary WSN deployments under the assumption that the constituent SNs follow the 3-mode, 2-par model.

3.5.2.1. 3-mode, 2-par SN Model

The 3-mode, 2-par SN model is built on the assumption that each SN in the WSN has two given probabilities of failure during T_m. The first one is the communication failure probability, denoted by λ_t^i which is defined as the probability that the wireless communication capability of SN i is lost due to transceiver hardware failure during T_m. The second probability is the sensing failure probability, denoted by λ_s^i which is the probability that the sensing capability of SN i is lost due to sensor hardware failure during T_m. We will

assume in this section that the battery and the processor of the SN are not subject to failure (later in Section 3.5.3 we will relax this assumption).

Accordingly, the events of communication and sensing failures are independent. This implies that a SN can have three modes of operation, namely *on*, *relay* and *off*. In the *on* mode, the SN has functioning sensor and transceiver hardware and hence can both sense and communicate wirelessly. In the *relay* mode, the SN has failed sensor hardware while its transceiver is functioning. In this mode, the SN will not be able to detect any events within its coverage range. However, it will still be able to communicate with its neighbors, i.e. act as a relay and hence contribute to the WSN functionality. On the other hand, if the SN transceiver fails, the SN becomes isolated from the network and hence is considered in the *off* mode, irrespective of the status of its sensor hardware.

3.5.2.2. Reliability Metric Derivation

Let the reliability of a WSN deployment $\boldsymbol{S}$ be denoted by $R(\boldsymbol{S})$. The reliability of the WSN deployment $\boldsymbol{S}$, $R(\boldsymbol{S})$, is defined as the probability that the WSN remains functional, in terms of coverage and connectivity, subject to two types of SN components failures during its intended mission time, T_m. To obtain a closed formula for $R(\boldsymbol{S})$ using the combinatorial approach to system reliability evaluation outlined in Section 3.4.2, let $\boldsymbol{X}_s$ denote the subset of SNs in $\boldsymbol{S}$ that have failed sensors and $\boldsymbol{X}_t$ denote the subset of SNs in $\boldsymbol{S}$ that have failed transceivers. We can express $R(\boldsymbol{S})$ as follows:

$$R(\boldsymbol{S}) = Prob(\boldsymbol{S}\ is\ functional) = \sum_{\boldsymbol{X}_s \subseteq \boldsymbol{S}} \sum_{\boldsymbol{X}_t \subseteq \boldsymbol{S}} [Prob(\boldsymbol{S}\ is functional \mid \boldsymbol{X}_s, \boldsymbol{X}_t).Prob(\boldsymbol{X}_s, \boldsymbol{X}_t)] \tag{3.8}$$

Equation (3.8) expresses $R(\boldsymbol{S})$ as the probability that $\boldsymbol{S}$ is functional, subject to all possible sensor and transceiver failure combinations of the SNs in $\boldsymbol{S}$ during T_m. The conditional probability of functionality is either equal to 1 or 0, depending on whether the WSN fulfills the two conditions of functionality stated in Definition 1 under a given sensor and transceiver failure combination, i.e. *network state*, represented by $\boldsymbol{X}_s$ and $\boldsymbol{X}_t$. Hence, the term $Prob(\boldsymbol{S}\ is functional \mid \boldsymbol{X}_s, \boldsymbol{X}_t)$ is equivalent to the *network structure function* which we will denote by $f(\boldsymbol{X}_s, \boldsymbol{X}_t)$. According to Definition 1, $f(\boldsymbol{X}_s, \boldsymbol{X}_t)$ can be expressed as follows:

$$f(\boldsymbol{X}_s, \boldsymbol{X}_t) = \begin{cases} 1, & if\ \ \boldsymbol{Z}_j \subseteq \boldsymbol{Y}_j \neq \phi\ \ \forall\, j = 1, \dots, m \\ 0, & otherwise \end{cases}. \tag{3.9}$$

To calculate the joint probability $Prob(\boldsymbol{X}_s, \boldsymbol{X}_t)$, i.e. the probability of a given network state represented $f(\boldsymbol{X}_s, \boldsymbol{X}_t)$, we can use the assumption that both types of SN failures are independent of each other. Hence, it can be expressed as follows,

$$f(\boldsymbol{X}_s, \boldsymbol{X}_t) = Prob(\boldsymbol{X}_s, \boldsymbol{X}_t) = Prob(\boldsymbol{X}_s) * Prob(\boldsymbol{X}_t). \tag{3.10}$$

Let $\boldsymbol{X}_s{}^c$ be the remainder of the SNs in $\boldsymbol{S}$ with functional sensors and $\boldsymbol{X}_t{}^c$ be the remainder of the sensors in $\boldsymbol{S}$ with functional transceivers. Hence, (3.8) can be rewritten as follows,

$$R(\boldsymbol{S}) = \sum_{\boldsymbol{X}_s \subseteq \boldsymbol{S}} \sum_{\boldsymbol{X}_t \subseteq \boldsymbol{S}} f(\boldsymbol{X}_s, \boldsymbol{X}_t).\, Prob(\boldsymbol{X}_s).\, Prob(\boldsymbol{X}_t) \tag{3.11}$$

$$= \sum_{\boldsymbol{X}_s \subseteq \boldsymbol{S}} \sum_{\boldsymbol{X}_t \subseteq \boldsymbol{S}} f(\boldsymbol{X}_s, \boldsymbol{X}_t).\, \prod_{i \in \boldsymbol{X}_s{}^c} \left(1 - \lambda_s^i\right).\, \prod_{i \in \boldsymbol{X}_s} \lambda_s^i .\, \prod_{i \in \boldsymbol{X}_t{}^c} (1 - \lambda_t^i).\, \prod_{i \in \boldsymbol{X}_t} \lambda_t^i$$

Examining (3.11) suggests that in the computation of the proposed reliability metric for most WSN deployments, there will be probability terms of very limited effect on the numerical value of reliability. If these terms are identified and discarded, then it is possible to calculate a lower bound of reliability that can serve as a good approximation of the exact reliability $R(\boldsymbol{S})$, with a smaller number of network structure function evaluations and hence lower computation time. The degree of approximation can be controlled by varying the number of discarded probability terms.

Let the set $\boldsymbol{F} = \{F_1, F_2, \ldots, F_{|\boldsymbol{F}|}\}$ contain all the SN failure combinations that can be tolerated by the WSN, where $|\boldsymbol{F}|$ denotes the number of these combinations. Let the term $Prob(F_q)$ for $q = 1, .., |\boldsymbol{F}|$ denote the probability of occurrence of the SN failure combination q in $\boldsymbol{F}$. We can express $Prob(F_q)$ by the following equation,

$$Prob(F_q) = \prod\nolimits_{i \in \boldsymbol{X}_{sq}^c} \left(1 - \lambda_s^i\right).\, \prod_{i \in \boldsymbol{X}_{sq}} \lambda_s^i .\, \prod\nolimits_{i \in \boldsymbol{X}_{tq}^c} (1 - \lambda_t^i).\, \prod_{i \in \boldsymbol{X}_{tq}} \lambda_t^i, \tag{3.12}$$

where $\boldsymbol{X}_{sq}$, $\boldsymbol{X}_{sq}{}^c$, $\boldsymbol{X}_{tq}$ and $\boldsymbol{X}_{tq}^c$, are the functional and dysfunctional SN sets which correspond to the SN failure combination F_k. Using (3.12), we can re-write the reliability metric expressed in (3.11) as follows,

$$R(\boldsymbol{S}) = \sum\nolimits_{q=1}^{|\boldsymbol{F}|} Prob(F_q). \tag{3.13}$$

Let the lower bound for $R(\boldsymbol{S})$ we wish to compute be denoted by $R_{lb}(\boldsymbol{S})$. Therefore, the expression for $R_{lb}(\boldsymbol{S})$ will include only a subset of the set of all the probability terms corresponding to the failure combinations in $\boldsymbol{F}$. If we assume that the combinations in $\boldsymbol{F}$ follow a descending order in terms of their probability of occurrence, then a cut-off value must be set to determine which probability terms are to be eliminated in the computation of $R_{lb}(\boldsymbol{S})$. This cut-off value will represent the smallest numerical value of a probability term that would be included in $R_{lb}(\boldsymbol{S})$. We will call that cut-off value the lower bound probability threshold, denoted by η_{lb}. Hence, we can express $R_{lb}(\boldsymbol{S})$ using the following equation:

$$R_{lb}(\boldsymbol{S}) = \sum\nolimits_{q=1}^{|\boldsymbol{F}|_{lb}} Prob(F_q), \tag{3.14}$$

where $|\boldsymbol{F}|_{lb} \leq |\boldsymbol{F}|$ is the number of probability terms include in $R_{lb}(\boldsymbol{S})$ such that $Prob(F_{|\boldsymbol{F}|_{lb}}) \geq \eta_{lb}$.

3.5.2.3. Reliability Metric Calculation

From the derived expression for the proposed reliability metric in (3.11), it is clear that computing $R(\boldsymbol{S})$ involves evaluating the WSN structure function for all possible combinations of SN failures, i.e. for all possible network states. This can pose a computational challenge since WSN deployments designed for real-world applications are often composed of tens or even hundreds of SNs, resulting in a huge number of possible

failure combinations, i.e. network states. To solve this computational problem, we make use of the following two properties of the WSN, $\boldsymbol{S}$, according to the adopted WSN model:

- Only tolerable SN failure combinations contribute to the value of $R(\boldsymbol{S})$ as expressed in (3.13). This means that the majority of the network states have null probabilities and hence they do not contribute to the value of $R(\boldsymbol{S})$.
- The WSN $\boldsymbol{S}$ has the property of being a monotone system [115]. This property implies the following. If the failure of a group of SNs' components causes $\boldsymbol{S}$ to fail, then the failure of any set which contains this group will also cause $\boldsymbol{S}$ to fail. For example, if we assume that the SNs s_1 and s_2 in the WSN depicted in Fig. 3.2 are both in the *off* mode while the remaining SNs are in the *on* mode, then it can readily be observed that this would cause $\boldsymbol{S}$ to fail since any phenomenon at target point t_1 cannot be detected or communicated to the sink node. This means that network states corresponding to this situation have a structure function of zero value. Using the monotone property, we can say that the network states that include the SNs s_1 and s_2 being in the *off* mode and s_4 being in the *relay* mode would also have a structure function value of zero without actually evaluating the function.

These two facts are used to construct a Breadth-First Search (BFS) algorithm to search for the tolerable SN failure combinations, i.e. generate the complete paths set and compute $R(\boldsymbol{S})$ using (3.13) for any WSN deployment $\boldsymbol{S}$. The algorithm can also compute the reliability lower bound $R_{lb}(\boldsymbol{S})$ for any given value of the parameter η_{lb}.

The structure of the algorithm is outlined in Table 3.1 and can be summarized as follows. In step 1, all required parameters for the computation of $R(\boldsymbol{S})$ are specified. If the exact value of $R(\boldsymbol{S})$ is to be computed, the lower bound probability threshold η_{lb} is set to zero. Step 2 sets the $stop\_flag$ parameter to zero. This parameter will flag the termination of the algorithm in case the lower bound reliability $R_{lb}(\boldsymbol{S})$ is being computed for a value of η_{lb} greater than zero. Step 2 also initializes $R(\boldsymbol{S})$ with the probability of the no SN failures network state. We are working under the assumption that the WSN is well designed and hence is functional under no failures. The sets $\boldsymbol{F}_s^k$, $\boldsymbol{F}_t^k$ and $\boldsymbol{F}^k$ are defined and initialized in step 3. $\boldsymbol{F}_s^k$ is the set that holds all the tolerable sensor failure combinations of length k assuming no transceiver failures while $\boldsymbol{F}_t^k$ holds all tolerable transceiver failure combinations of length k assuming no sensor failures. The set $\boldsymbol{F}^k$ is defined as the set that holds all the tolerable components failures of length k. The algorithm then proceeds by evaluating the failure combinations which can be tolerated by the WSN for an increasing value of , i.e. proceeds in a breadth-first approach, in steps 4 to 7. For each value of k, the probabilities of the events corresponding to the failure combinations in the set $\boldsymbol{F}^k$ are computed and then re-ordered in a descending order accordingly. This step is carried out to make sure that all discarded probability terms in the calculation of $R_{lb}(\boldsymbol{S})$ are indeed of a value smaller than η_{lb}. If η_{lb} is greater than zero, the algorithm will terminate when a probability term less than η_{lb} is calculated corresponding to a failure combination in $\boldsymbol{F}^k$ at the current value of k. If η_{lb} is set to zero to calculate the exact value of $R(\boldsymbol{S})$, then the algorithm terminates when all tolerable failure combinations $(\boldsymbol{X}_s, \boldsymbol{X}_t)$ are evaluated and any additional component failure results in $f(\boldsymbol{X}_s, \boldsymbol{X}_t)$=0.

Table 3.1 Pseudo-code for the proposed algorithm for calculating the reliability of a WSN assuming SNs follow a 3-mode, 2-par model

Step	Algorithm for computing reliability $R(\boldsymbol{S})$ lower bound reliability R_{lb}
1.	Set all parameters ($\boldsymbol{S}$, $\{t_j\}$, λ_s^i, λ_t^i and η_{lb} for $j = 1, \dots, m$ and $i = 1, \dots, n$)
2.	Initialize: set $stop\_flag$ = 0 and $R(\boldsymbol{S}) = Prob([\boldsymbol{X}_s = \{\phi\}, \boldsymbol{X}_t = \{\phi\}])$
3.a.	Let k be the number of failed components. Initialize $k = 1$.
3.b.	Let $\boldsymbol{F}_s^k$ and $\boldsymbol{F}_t^k$ be the sets of k −combinations of failed sensors and transceivers that $\boldsymbol{S}$ can tolerate respectively. Initialize $\boldsymbol{F}_s^k = \{\phi\}$ and $\boldsymbol{F}_t^k = \{\phi\}$.
3.c.	Let $\boldsymbol{F}^k$ be the set of all k −combinations of failed components that $\boldsymbol{S}$ can tolerate. Initialize $\boldsymbol{F}^k = \{\phi\}$.
4.a.	For $i = 1, \dots, n$ - Let $\boldsymbol{X}_s = \{i\}$ and $\boldsymbol{X}_t = \{\phi\}$ - If $f(\boldsymbol{X}_s, \boldsymbol{X}_t) = 1 \rightarrow \boldsymbol{F}_s^k = \boldsymbol{F}_s^k \cup [\boldsymbol{X}_s, \boldsymbol{X}_t = \{\phi\}]$ End For
4.b.	For $i = 1, \dots, n$ - Let $X_s = \{\phi\}$ and $X_t = \{i\}$ - If $f(\boldsymbol{X}_s, \boldsymbol{X}_t) = 1 \rightarrow \boldsymbol{F}_t^k = \boldsymbol{F}_t^k \cup [\boldsymbol{X}_s = \{\phi\}, \boldsymbol{X}_t]$ End For
4.c.	$\boldsymbol{F}^k = \boldsymbol{F}^k \cup \boldsymbol{F}_s^k \cup \boldsymbol{F}_c^k$
5.a.	Let $F_l^k \in \boldsymbol{F}^k$, $l = 1, \dots, \lvert\boldsymbol{F}^k\rvert$. Calculate $Prob(F_l^k)$ $\forall l$. Rearrange $\boldsymbol{F}^k$ accordingly in descending order.
5.b.	For $l = 1, \dots, \lvert\boldsymbol{F}^k\rvert$ - If $Prob(F_l^k) < \eta_{lb} \rightarrow stop\_flag = 1 \rightarrow$ break For → go to 9 - Else $R(\boldsymbol{S}) = R(\boldsymbol{S}) + Prob(F_l^k)$ End For
6.	While ($stop\_flag \neq 0$ and $\boldsymbol{F}^k \neq \{\phi\}$) $k = k + 1$. $\boldsymbol{F}_s^k = \{\phi\}$, $\boldsymbol{F}_t^k = \{\phi\}$ and $\boldsymbol{F}^k = \{\phi\}$.
7.a.	Let $F_{sl}^{k-1} \in \boldsymbol{F}_s^{k-1}$, $l = 1, \dots, \lvert\boldsymbol{F}_s^{k-1}\rvert$
7.b	For $l = 1, \dots, \lvert\boldsymbol{F}_s^{k-1}\rvert$ For $i = 1, \dots, \lvert\boldsymbol{F}_s^1\rvert$ - Let $\boldsymbol{X}_s = \{\boldsymbol{F}_{sl}^{k-1}, i\}$ and $\boldsymbol{X}_t = \{\phi\}$ - If $f(\boldsymbol{X}_s, \boldsymbol{X}_t) = 1 \rightarrow \boldsymbol{F}_s^k = \boldsymbol{F}_s^k \cup [\boldsymbol{X}_s, \boldsymbol{X}_t = \{\phi\}]$ End For End For
7.c.	Let $F_{tl}^{k-1} \in \boldsymbol{F}_t^{k-1}$, $k = 1, \dots, \lvert\boldsymbol{F}_t^{k-1}\rvert$
7.d.	For $l = 1, \dots, \lvert\boldsymbol{F}_t^{k-1}\rvert$ For $i = 1, \dots, \lvert\boldsymbol{F}_t^1\rvert$ - Let $\boldsymbol{X}_t = \{F_{tl}^{k-1}, i\}$ and $\boldsymbol{X}_s = \{\phi\}$ and - If $f(\boldsymbol{X}_s, \boldsymbol{X}_t) = 1 \rightarrow \boldsymbol{F}_t^k = \boldsymbol{F}_t^k \cup [\boldsymbol{X}_s = \{\phi\}, \boldsymbol{X}_t]$ End For End For
7.e.	$\boldsymbol{F}^k = \boldsymbol{F}^k \cup \boldsymbol{F}_s^k \cup \boldsymbol{F}_t^k$
7.f.	For $l = 1, \dots, (m-1)$ For $l_s = 1, \dots, \lvert\boldsymbol{F}_s^{k-l}\rvert$ For $l_t = 1, \dots, \lvert\boldsymbol{F}_c^l\rvert$ - Let $\boldsymbol{X}_s = \{F_s^{k-l}\}$, $\boldsymbol{X}_t = \{F_t^l\}$ - If $f(\boldsymbol{X}_s, \boldsymbol{X}_t) = 1 \rightarrow \boldsymbol{F}^k = \boldsymbol{F}^k \cup [\boldsymbol{X}_s, \boldsymbol{X}_t]$ End For End For End For
8.	Repeat step 5.
9.	Print $R(\boldsymbol{S})$

3.5.3. Reliability Metric Formulation for the 3-mode, 4-par SN Model

In this section, we re-drive the proposed WSN reliability metric of Section 3.5.2 under the assumption that the constituent SNs follow the 3-mode, 4-par model.

3.5.3.1. 3-mode, 4-par SN Model

This model assumes that SNs which are characterized by four different probabilities of failure during T_m, namely, the sensor, transceiver, processor and battery failures. These failure probabilities can be estimated through a standard reliability prediction test provided by the SN vendor or through reliability testing techniques [97].

Since each of the four components can either function or fail, i.e. be in an *on* or *off* state, an SN can theoretically have 2^4 possible states. To describe these states, let the binary variables x_s, x_t, x_p and x_b be the state indicators of the sensor, transceiver, processor and battery, respectively, of an SN, as defined in (3.2) in Section 3.4.2. Hence, an SN state x is described using a tuple of these four variables $\{x_s, x_t, x_p, x_b\}$. These variables are not statistically independent; the sensor and transceiver cannot possibly function if either the processor or the battery fails. Therefore, some of the SN states are practically impossible and hence their probability of occurrence is zero.

To calculate the probability of occurrence of the other possible states, let λ_s, λ_t, λ_p and λ_b be the probabilities of failure of the sensor, transceiver, processor and battery, respectively. It should be noted that only unrecoverable hardware failures of these four SN components are considered here, i.e. temporary failures/malfunctions are not considered. It should also be noted that the estimated probability of failure for any given device or hardware component is obtained regardless of the failure of any other device or component. Hence, λ_s and λ_t are actually the probability of failure of the sensor and transceiver *conditioned* on the event that the component is properly controlled (i.e. processor is functional) and powered (i.e. battery is functional). Similarly, λ_p is the probability of failure of the processor *conditioned* on the event that the battery is functional, where as λ_b is the unconditional probability that the SN power unit or battery fails during T_m. According to the above definitions, the probability of an SN state can be given by the following equations:

$$Prob(x) = Prob(x_s, x_t, x_p, x_b) = Prob(x_s, x_t / x_p, x_b).\, Prob(x_p, x_b) = Prob(x_s / x_p, x_b).\, Prob(x_t / x_p, x_b).\, Prob(x_p | x_b).\, Prob(x_b) \tag{3.15}$$

Equation (3.15) makes use of the fact that the states of the sensor and the transceiver are independent when conditioned on the states of the processor and battery. Fig. 3.3 illustrates the SN's states which have a non-zero probability. It is straightforward to verify that the sum of the probabilities of these states is equal to unity. There are two SN states at which the SN is of use to the WSN. The first state is described by the tuple {1,1,1,1}, at which all four components are functional and the SN can both sense its surroundings and communicate wirelessly. This state corresponds to the *on* mode of operation in which the SN is fully functional. The second state is described by the tuple {0,1,1,1} at which only the sensor(s) failed and the SN can only communicate wirelessly, i.e. acts as a relay node. This state corresponds to the *relay* mode of operation in which the SN is partially functional. In all the practically possible remaining states the SN does not serve the network and hence a SN in these states is considered to be in the *off* mode of operation.

3.5.3.2. Reliability Metric Derivation

The reliability of the WSN deployment $\boldsymbol{S}$, denoted by $R(\boldsymbol{S})$, is defined as the probability that the WSN remains functional, in terms of coverage and connectivity, subject to four types of SN components failures during its intended mission time, T_m. We follow the combinatorial approach outlined in Section 3.4.2 to derive $R(\boldsymbol{S})$.

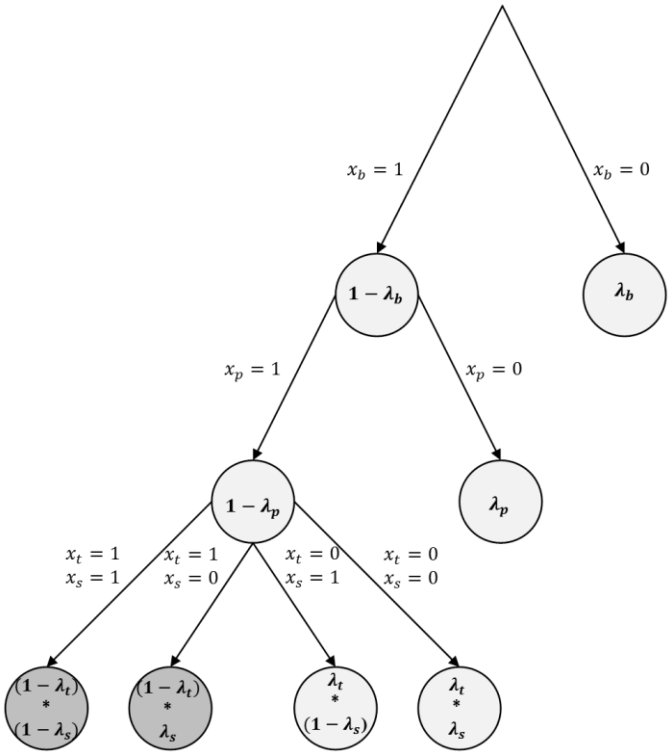


Fig. 3.3 SN states: the paths from the top node to a bottom node correspond to the SN states with non-zero probability. The probability of a path, i.e. the probability of a state, is the product of the probabilities in the associated transitions. The SN *relay* state and the *on* state are both marked by a dark shade of grey.

To define the states of $\boldsymbol{S}$, let $\boldsymbol{X}_s, \boldsymbol{X}_t, \boldsymbol{X}_p$ and $\boldsymbol{X}_b$ be the subsets of SNs in $\boldsymbol{S}$ that have failed sensors, transceivers, processors and batteries, respectively. Hence, a state of the WSN $\boldsymbol{S}$ is described by the tuple $\boldsymbol{\pi} \equiv \{\boldsymbol{X}_s, \boldsymbol{X}_t, \boldsymbol{X}_p, \boldsymbol{X}_b\}$, where $\boldsymbol{X}_s, \boldsymbol{X}_t, \boldsymbol{X}_p, \boldsymbol{X}_b \subseteq \boldsymbol{S}$. Therefore, each state $\boldsymbol{\pi}$ is associated with a unique combination of SN components' failures. To calculate the probability of occurrence of a given state, $\boldsymbol{\pi}$, the corresponding state $x_i(\boldsymbol{\pi})$ of each individual SN $s_i \in \boldsymbol{S}$ must be identified. Assuming the components belonging to different SNs fail independently, $Prob(\boldsymbol{\pi})$ can be expressed by:

$$Prob(\boldsymbol{\pi}) = Prob\big(\boldsymbol{X}_s, \boldsymbol{X}_t, \boldsymbol{X}_p, \boldsymbol{X}_b\big) = \prod_{i=1}^{N} Prob(x_i(\boldsymbol{\pi})) \tag{3.16}$$

Table 3.2 lists the different values of the probability of an individual SN state $x_i(\boldsymbol{\pi})$ for a given network state $\boldsymbol{\pi}$ based on the SN states illustrated in Fig. 3.3. Let $\boldsymbol{\Pi}$ be the set of all possible states of $\boldsymbol{S}$. Based on the definition provided in Section 3.5.1, the structure function of $\boldsymbol{S}$ can be expressed as follows:

$$f(\boldsymbol{\pi}) = \begin{cases} 1, & if\ \ \boldsymbol{Z}_j \subseteq \boldsymbol{Y}_j \neq \phi \quad \forall\, j = 1, \dots, m \\ 0, & otherwise \end{cases} \tag{3.17}$$

Similar to (3.4), we can now express the reliability of the WSN $\boldsymbol{S}$ as follows:

$$R(\boldsymbol{S}) = Prob(\boldsymbol{S}\ is\ functional\ during\ T_m) = \sum_{\boldsymbol{\pi}\epsilon\boldsymbol{\Pi}} [f(\boldsymbol{\pi}).Prob(\boldsymbol{\pi})]$$

$$= \sum_{\boldsymbol{X_s}\subseteq\boldsymbol{S}} \sum_{\boldsymbol{X_t}\subseteq\boldsymbol{S}} \sum_{\boldsymbol{X_p}\subseteq\boldsymbol{S}} \sum_{\boldsymbol{X_b}\subseteq\boldsymbol{S}} \left[f(\boldsymbol{X}_s, \boldsymbol{X}_t, \boldsymbol{X}_p, \boldsymbol{X}_b). \prod_{i=1}^{N} Prob(x_i\,(\boldsymbol{\pi})) \right] \tag{3.18}$$

Table 3.2 Evaluation of the probability of the corresponding individual SN states for a given WSN state $= \{\boldsymbol{X}_s, \boldsymbol{X}_t, \boldsymbol{X}_p, \boldsymbol{X}_b\}$, where "true" and "false" are denoted by 1 and 0 respectively and λ_s^i, λ_t^i, λ_p^i, λ_b^i are the probabilities of failure of the four main components

$s_i \in \boldsymbol{X}_s$	$s_i \in \boldsymbol{X}_t$	s_i	$s_i \in \boldsymbol{X}_b$	$Prob(\, x_i(\boldsymbol{\pi}))$
0	0	0	0	$(1\text{-}\lambda_s^i)(1\text{-}\lambda_t^i)(1\text{-}\lambda_p^i)(1\text{-}\lambda_b^i\ )$
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	$(1\text{-}\lambda_s^i)\lambda_t^i\ (1\text{-}\lambda_p^i)(1\text{-}\lambda_b^i\ )$
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	$\lambda_s^i(1\text{-}\lambda_t^i)(1\text{-}\lambda_p^i)(1\text{-}\lambda_b^i\ )$
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	$\lambda_s^i\lambda_t^i\ (1\text{-}\lambda_p^i)(1\text{-}\lambda_b^i\ )$
1	1	0	1	0
1	1	1	0	$\lambda_p^i(1\text{-}\lambda_b^i\ )$
1	1	1	1	λ_b^i

3.5.3.3. Reliability Metric Calculation

Similar to the reliability metric expressed in (3.11), evaluating $R(\boldsymbol{S})$ using the expression in (3.18) involves evaluating the structure function of the network, $f(\boldsymbol{\pi})$, for all possible states of $\boldsymbol{S}$, i.e. for all $\boldsymbol{\pi}\epsilon\boldsymbol{\Pi}$. To calculate $R(\boldsymbol{S})$ using 3.18 in a time efficient manner, we propose a BFS algorithm that generates the complete paths set of $\boldsymbol{S}$, denoted by $\boldsymbol{\Pi}_1$. The algorithm is founded on the same two properties discussed in Section 3.5.2.3. The general structure of the proposed search algorithm is illustrated in Fig. 3.4. The pseudo-code of the algorithm, which provides execution details, is given in Table 3.3.

The structure of the algorithm can be summarized in the following steps. In step 1, all the required parameters for the calculation of $R(\boldsymbol{S})$ are specified as inputs. This includes the two dimensional RoI layout, the positions of the target locations within the RoI provided by the set of target points $\boldsymbol{T} = \{t_j\}$, the positions of the deployed SNs provided by $\boldsymbol{S} = \{s_i\}$, the types of the deployed SNs including their coverage profiles and wireless communication ranges and the probabilities of failure of the SN components associated with each SN type. We assume here that the sink node can be at any fixed arbitrary position in the targeted RoI.

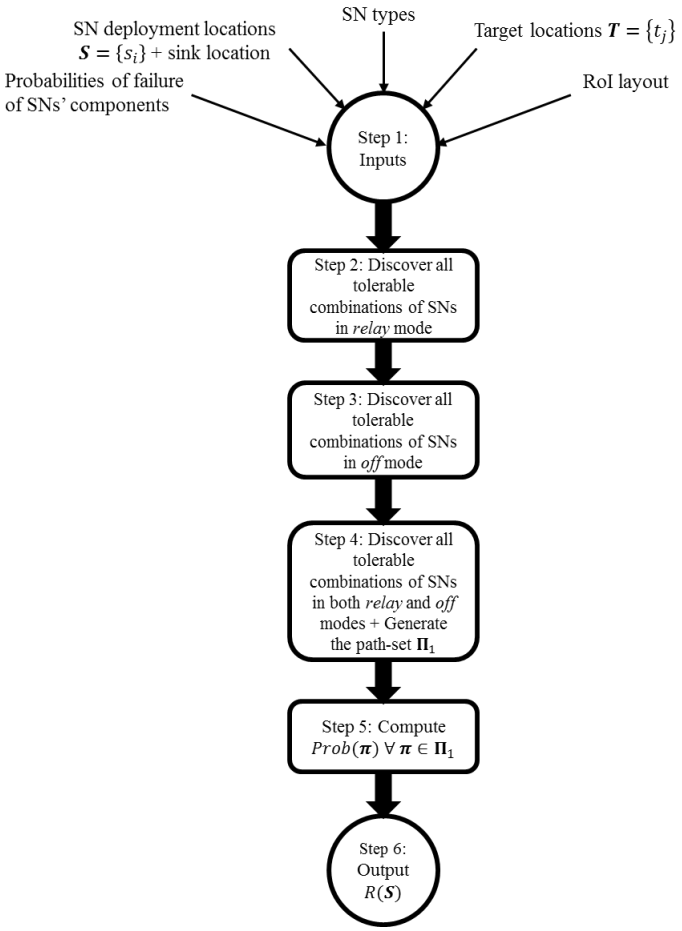


Fig. 3.4 The structure of the proposed algorithm for evaluating the WSN reliability $R(\boldsymbol{S})$.

We initialize the value of $R(\boldsymbol{S})$ with the probability of the network state $\boldsymbol{\pi}$ which corresponds to all the deployed SNs being in the on mode. Since this network state is an obvious path of $\boldsymbol{S}$, we also initialize the network path set $\boldsymbol{\Pi}_1$ with this state as expressed in 1.c. in Table 3.3.

In step 2, the algorithm searches for all the combinations of SNs that can be in the *relay* mode without compromising the functionality of $\boldsymbol{S}$, assuming the remainder of the deployed SNs are in the *on* mode. These SN combinations are referred to as the "tolerable combinations of SNs in the *relay* mode". This means that for the network states corresponding to these SN combinations, the structure function expressed in (3.17) is equal to unity. To perform this search, we define $\boldsymbol{F}_r^k$ as the set that holds the tolerable combinations of SNs in *relay* mode of length k starting with $k = 1$ as expressed in 2.a - 2.c. in Table 3.3. For example, consider the WSN depicted in Figure 3.2. The set of single tolerable SNs in the

Table 3.3 Pseudo-code for the proposed algorithm for calculating the reliability of a WSN assuming SNs follow a 3-mode, 4-par model

Step	Algorithm for computing WSN reliability $R(\boldsymbol{S})$
1.a.	Set all parameters ($\boldsymbol{S} = \{s_i\}$, $\boldsymbol{T} = \{t_j\}$, types of SNs, sink location, λ_s^i, λ_t^i, λ_p^i and λ_b^i for $i = 1, \dots, n$ and $j = 1, \dots, m$)
1.b.	Initialize $R = Prob(\boldsymbol{\pi} \mid s_i \in \boldsymbol{S}$ is in *on* mode $\forall\ i = 1, \dots, n)$
1.c.	Initialize $\boldsymbol{\Pi}_1 = \{(\boldsymbol{\pi} \mid s_i \in \boldsymbol{S}$ is in *on* mode $\forall\ i = 1, \dots, n)\}$
2.a.	Let k be the number of SNs in *relay* mode. Initialize $k = 1$.
2.b.	Let $\mathcal{F}_r^k$ be a $k-$combination of SNs in *relay* mode. Let $\boldsymbol{F}_r^k$ be the set of $k-$combinations of SNs in *relay* mode that $\boldsymbol{S}$ can tolerate. Initialize $\mathcal{F}_r^k = \boldsymbol{F}_r^k = \{\phi\}$.
2.c.	For $i = 1, \dots, n$ - Let s_i be in relay mode, i.e. $\mathcal{F}_r^k = \{s_i\}$ - Evaluate $f(\boldsymbol{\pi} \mid \mathcal{F}_r^k)$ using (3.17) - If $f(\boldsymbol{\pi} \mid \mathcal{F}_r^k) = 1 \rightarrow \boldsymbol{F}_r^k = \boldsymbol{F}_r^k \cup \mathcal{F}_r^k$ End For loop
2.d.	While $\boldsymbol{F}_r^k \neq \{\phi\} \quad \rightarrow \quad k = k + 1$, Let $\boldsymbol{F}_{rl}^{k-1} \in \boldsymbol{F}_r^{k-1}$, $\mathcal{F}_r^k = \boldsymbol{F}_r^k = \{\phi\}$
2.e.	For $l = 1, \dots, \lvert \boldsymbol{F}_r^{k-1} \rvert$ and $i = 1, \dots, \lvert \boldsymbol{F}_r^1 \rvert$ - Let $\mathcal{F}_r^k = \{ \boldsymbol{F}_{rl}^{k-1}, s_i \}$ - Evaluate $f(\boldsymbol{\pi} \mid \mathcal{F}_r^k)$ using (3.17) - If $f(\boldsymbol{\pi} \mid \mathcal{F}_r^k) = 1 \rightarrow \boldsymbol{F}_r^k = \boldsymbol{F}_r^k \cup \mathcal{F}_r^k$
2.f.	End For loops, End While loop
3.a.	Let k be the number of SNs in *off* mode. Initialize $k = 1$.
3.b.	Let $\mathcal{F}_o^k$ be a $k-$combination of SNs in *off* mode. Let $\boldsymbol{F}_o^k$ be the set of $k-$combinations of SNs in *off* mode that $\boldsymbol{S}$ can tolerate. Initialize $\mathcal{F}_o^k = \boldsymbol{F}_o^k = \{\phi\}$.
3.c.	Repeat step 2.c. for *off* mode, i.e. $\mathcal{F}_o^k = \{s_i\}$
3.d.	While $\boldsymbol{F}_o^k \neq \{\phi\} \quad \rightarrow \quad k = k + 1$, Let $F_{ol}^{k-1} \in \boldsymbol{F}_o^{k-1}$, $\mathcal{F}_o^k = \boldsymbol{F}_o^k = \{\phi\}$
3.e.	Repeat 2.e. using $\boldsymbol{F}_{ol}^{k-1}$ and $\mathcal{F}_o^k$ to get $\boldsymbol{F}_o^k$, $i = 1, \dots, \lvert \boldsymbol{F}_o^1 \rvert$
3.f.	End For loops, End While loop
4.a.	Let $\mathcal{F}_r$ and $\mathcal{F}_o$ be a combination of SNs in *relay* and *off* modes respectively. Let $\boldsymbol{F}_r$ and $\boldsymbol{F}_o$ be the sets of all combinations of SNs of in *relay* and *off* mode that that $\boldsymbol{S}$ can tolerate respectively. Let $F_{rl_r} \in \boldsymbol{F}_r$ and $F_{ol_o} \in \boldsymbol{F}_o$
4.b.	For $l_r = 1, \dots, \lvert \boldsymbol{F}_r \rvert$ and $l_o = 1, \dots, \lvert \boldsymbol{F}_o \rvert$ - Let $\mathcal{F}_r = \boldsymbol{F}_{rl_r}$ and $\mathcal{F}_o = \boldsymbol{F}_{ol_o}$ - Evaluate $f(\boldsymbol{\pi} \mid \mathcal{F}_r, \mathcal{F}_o)$ using (3.17) - If $f(\boldsymbol{\pi} \mid \mathcal{F}_r, \mathcal{F}_o) = 1 \rightarrow \Pi_1 = \Pi_1 \cup \boldsymbol{\pi}$ End For loops
5.a.	Let $\boldsymbol{\pi}_l \in \boldsymbol{\Pi}_1$
5.b.	For $l = 1, \dots, \lvert \boldsymbol{\Pi}_1 \rvert$ - $R(\boldsymbol{S}) = R(\boldsymbol{S}) + Prob(\boldsymbol{\pi}_l)$ End For loop
6.	Output: $R(\boldsymbol{S})$

relay mode will be given by $\boldsymbol{F}_r^1 = \{ \{s_1\}, \{s_2\}, \{s_3\}, \{s_4\}, \{s_5\} \}$. The algorithm then proceeds with the search for an increasing value of k as expressed in 2.d - 2.f. in Table 3.3. For example, the combination $\{s_1, s_3\}$ belongs to $\boldsymbol{F}_r^2$ while $\{s_1, s_2\}$ does not. This search continues until the algorithm reaches a value of k which results in an empty $\boldsymbol{F}_r^k$, i.e. $\boldsymbol{F}_r^k = \{\emptyset\}$. The set of all tolerable combinations of different lengths of SNs in *relay* mode is denoted $\boldsymbol{F}_r$.

In step 3, the algorithm searches for all the combinations of SNs that can be in the *off* mode without compromising the functionality of $\boldsymbol{S}$, assuming the remainder of the SNs is in the *on* mode, i.e. tolerable combinations of SNs in the *off* mode. The search follows the same

procedure in step 2. Note that if the network cannnot tolerate a SN being in the *relay* mode, it follows that it can not tolerate it in the *off* mode. This observation reduces the number of structre function evaluation in step 3. We define $\boldsymbol{F}_o^k$ as the set that holds the tolerable combinations of SNs in the *off* mode of length k. Using the same example WSN in Fig. 3, $\boldsymbol{F}_o^1 = \{\{s_2\},\{s_3\},\{s_4\},\{s_5\}\}$. The combination $\{s_4, s_5\}$ belongs to $\boldsymbol{F}_o^2$ while $\{s_2, s_5\}$ does not. The set of all tolerable combinations of different lengths of SNs in the *off* mode is denoted $\boldsymbol{F}_o$.

In step 4, the algorithm uses the sets $\boldsymbol{F}_r$ and $\boldsymbol{F}_o$ to discover all the pairs of combinations of SNs that can be in the *relay* and *off* modes simultaneously without compromising the functionality of $\boldsymbol{S}$, assuming the remainder of the SNs is in the *on* mode. For example, the combination $\{s_1, s_3\}$ can be in the *relay* mode while $\{s_5\}$ can be in the *off* mode simultaneously without causing the WSN depicted in Fig. 3.2 to fail. Each of the discovered pairs of combinations corresponds to one or more distinct network path and hence the complete paths set $\boldsymbol{\Pi}_1$ is updated accordingly as expressed in 4.b in Table 3.3. In step 5, the probabilities of the network paths in $\boldsymbol{\Pi}_1$ are calculated using (3.16) and Table 3.2. Finally, the reliability of the given WSN $R(\boldsymbol{S})$ is calculated using (3.18) and given as an output in step 6.

3.6. Case Study

In this section, we apply the proposed reliability metric in both its versions, as expressed in (3.11) and (3.18), to evaluate the reliability of a surveillance WSN deployments designed to cover part of an international airport terminal. We compare between each version of the proposed reliability metric and the existing reliability metric proposed in [107] to highlight the significance of using the 3-mode SN model on the accuracy of reliability evaluation. We also evaluate the computational efficiency of the developed BFS algorithm used to calculate the proposed reliability metric. Finally, we compare between both versions of the proposed reliability metric in terms of the accuracy of reliability evaluation.

3.6.1. Experimental Set-up

Consider the layout of an international airport terminal shown in Fig. 3.5 in which surveillance WSN is to be designed to cover part of it. Target points, marked on the Fig. 3.5 in red, represent the vital locations that need to be placed under image/video surveillance such as arrival checkpoints, entrances and staircases. The sink node to which all SNs in the WSN should be connected is marked in black.

To obtain test deployments of the WSN for different number of target points, we use the iVLGA proposed in [75] and discussed in Section 2.4.1 in Chapter 2. This optimization algorithm is designed to obtain cost-optimized deployments for heterogeneous WSNs that provide coverage for all designated target points in the RoI, i.e. providing full-coverage of the set $\boldsymbol{T} = \{t_j\}$ for $j = 1, \ldots, m$. However, since a well-designed surveillance WSN should be functional in terms of coverage and connectivity, we modified the iVLGA in [75] to add network connectivity to the design objectives. To achieve this we modify the fitness function of the algorithm expressed in (2.27), which is used to evaluate the fitness of the candidate deployments or chromosomes in [75], as follows:

$$f(c(l)) = -\left(\sum_{i=1}^{l} p_i + w_1 * (m - cov) + w_2 * con\_test\right) \tag{3.19}$$

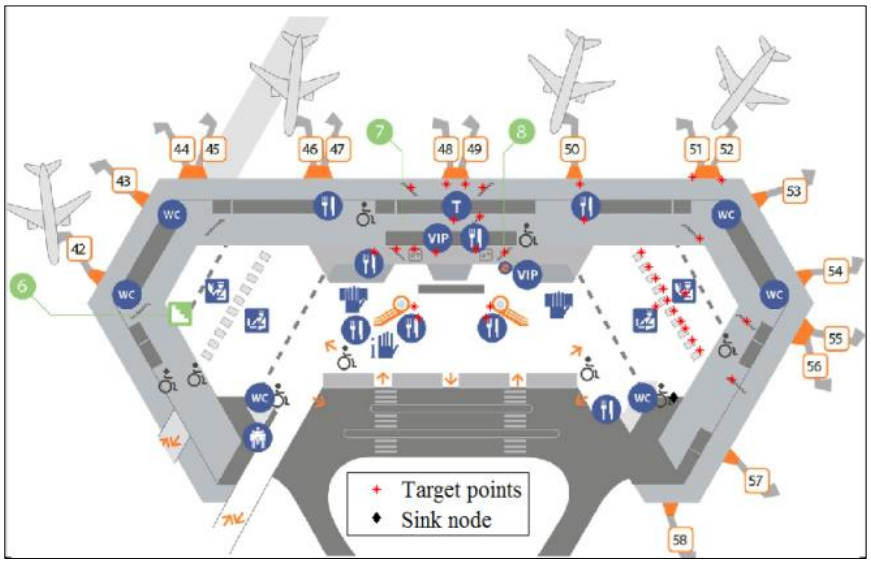


Fig. 3.5 Schematic of an international airport terminal with the marked positions of the target points, possible deployment points and sink node of layout 5 in Table 3.5.

where $\sum_{i=1}^{l} p_i$ is the total cost of the deployment $c(l)$, cov is the number of target points that are covered by $c(l)$, con_test is a binary variable that is equal to unity when $c(l)$ is a disconnected deployment (i.e. has isolated SNs from the sink node) and zero otherwise, w_1 is the penalty imposed on the fitness for failing to cover a single target point and w_2 is a penalty for violating the connectivity constraint. For further details on the VLGA and the settings of its parameters, we refer the reader to the study in [75].

To demonstrate the ability of the proposed metric to evaluate the reliability of heterogeneous deployments, we assume that there are two types of image/video SNs available for the deployment of the WSN with different operational parameters.

3.6.2. Results and Discussion for the 3-mode, 2-Par SN model

In this section we compare the proposed reliability metric using the 3-mod, 2-par SN model as expressed in (3.11) to the existing reliability metric in [107] which uses the conventional 2-mode SN model. The operational parameters of the two types of SNs used in this experiment are listed in Table 3.4. Although the exact reliability figures for commercial SNs such as Tmote2 and Iris nodes are not publicly available, we estimated the given probabilities of failure using the reliability figures available for Texas Instrument CC2420 IEEE 802.15.4 transceiver [116] as a reference point, assuming a WSN mission time of five years. We also used the fact that sensor hardware is the SN component most prone to failure [8]. Using the modified iVLGA, we obtain five WSN deployments for five sets of target points of different sizes. Table 3.5 lists the data of the five deployments, including the number of target points m, the number of deployed SNs of type 1 and type 2, denoted by n_1, and n_2, respectively, and the total number of deployed SNs in the given deployment denoted by n.

For each deployment in Table 3.5, the reliability is assessed using three different methods. The first and second methods apply the proposed reliability metric $R(\boldsymbol{S})$ expressed in (3.11) and its lower bound $R_{lb}(\boldsymbol{S})$ expressed in (3.14) for $\eta_{lb} = 10^{-3}$ respectively. The third method assesses the reliability using the reliability metric proposed in [107]. For a fair comparison, we use the same WSN functionality definition (in terms of both network coverage and connectivity) expressed in (3.9) for all the three methods. Since the metric

in [107] adopts the conventional two-mode SN model in which a given SN is either in a fully functional or failed state, SNs cannot contribute to the WSN functionality as relays. In this case, the corresponding probability of failure of a given SN is equal to $1-\left(1-\lambda_s^i\right)(1-\lambda_t^i)$.

Fig. 3.6 shows the performance of the three reliability assessment methods in the computed reliability value for the five deployments in Table 3.5. Fig. 3.7 shows the number of tolerable SN failure combinations contributing to the value of reliability (i.e. $|\boldsymbol{F}|$) obtained from the three methods. As can be observed from Fig. 3.6, the reliability values computed using the metric in [107] are significantly lower than those computed using the proposed metric and its lower bound for all the deployments of Table 3.5, with the difference reaching around 5% for the fifth deployment. This behavior can be attributed to the fact that the conventional 2-mode SN model adopted in [107] does not take into account the ability of an SN with a failed sensor to contribute to the functionality of the WSN as a relay. Consequently, the reliability of a given deployment is underestimated. The difference between both metrics in the computed reliability value is directly proportional to the number of SNs with redundant coverage, i.e. number of SNs that can be in the *relay* mode without compromising the functionality of the network. Fig. 3.6 and Fig. 3.7 show that the reliability values computed using the three reliability assessment methods decrease steadily with the increase in the size of the WSN, i.e. the increase in the number of deployed SNs n, while the value of $|\boldsymbol{F}|$ increases. This behavior can be explained as follows. For the deployments in Table 3.5, as n increases, the number of redundant SNs (both complete redundancy and coverage redundancy) increases as well. This is because it is more likely for a larger WSN to be able to tolerate the failure of a few SNs than a smaller one, especially in terms of coverage, given that both deployments are cost-optimized (i.e. characterized by a low level of SN redundancy). The value of $|\boldsymbol{F}|$, in turn, increases with the level of SN redundancy, especially for the proposed metric $R(\boldsymbol{S})$ which reflects the SN coverage redundancy. However, the rate of increase in the number of redundant SNs is less than the rate of increase of n itself. This means that the *relative* SN redundancy level decreases with the increase in n and hence the reliability decreases.

Fig. 3.6 also shows that $R(\boldsymbol{S})$ and $R_{lb}(\boldsymbol{S})$ are very close in value. The difference between them increases slightly with the increase of n, but does not exceed 2% for the fifth deployment. This is because the contribution of the omitted probability terms from the expression of $R_{lb}(\boldsymbol{S})$ to the exact value of $R(\boldsymbol{S})$ is very small in value compared to the included terms. From Fig. 3.7 we can observe that as n increases, the value $|\boldsymbol{F}|$ increases as well as the number of tolerable SN failure combinations with probabilities less than η_{lb}.

Table 3.4 Parameters of the SN types used in the deployments listed in Table 3.5

	FoV	r_s	r_c	λ_S	λ_t	**Price**(\$)
Type 1	90°	20 m	40 m	1.0×10^{-2}	5.0×10^{-3}	150
Type 2	60°	30 m	40 m	2.0×10^{-2}	1.5×10^{-2}	100

Table 3.5 Data of the obtained deployments for the case-study surveillance WSN

	m	n_1	n_2	n	C \$
Deployment 1	15	0	11	11	1100
Deployment 2	20	5	13	18	2050
Deployment 3	25	6	17	23	2600
Deployment 4	30	11	17	28	3350
Deployment 5	35	12	22	34	4000

Therefore, the difference between the number of terms which goes into the calculation of $R(\boldsymbol{S})$ and $R_{lb}(\boldsymbol{S})$ increases as well. Hence, the difference between the values of $R(\boldsymbol{S})$ and $R_{lb}(\boldsymbol{S})$ for a given WSN deployment is inversely proportional to the threshold η_{lb} and is directly proportional to the level SN redundancy in that deployment. This means that for cost-optimized WSN deployments characterized by low SN redundancy level, the lower bound can serve as a good approximation for the exact reliability. On the other hand, if the level of redundancy is high, the neglected terms increase in number and causes the difference between $R(\boldsymbol{S})$ and $R_{lb}(\boldsymbol{S})$ to become higher.

Fig. 3.8 shows a comparison between the computation time incurred by the three reliability assessment methods for the five deployments in Table 3.5. It can be readily observed that the computation time incurred by $R(\boldsymbol{S})$ is notably higher than that incurred by

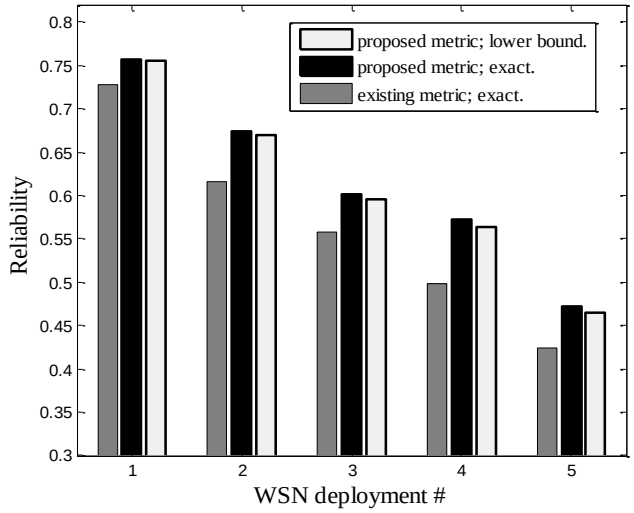


Fig. 3.6 Comparison among the existing reliability metric in [107], the proposed metric $R(\boldsymbol{S})$ and its lower bound $R_{lb}(\boldsymbol{S})$.

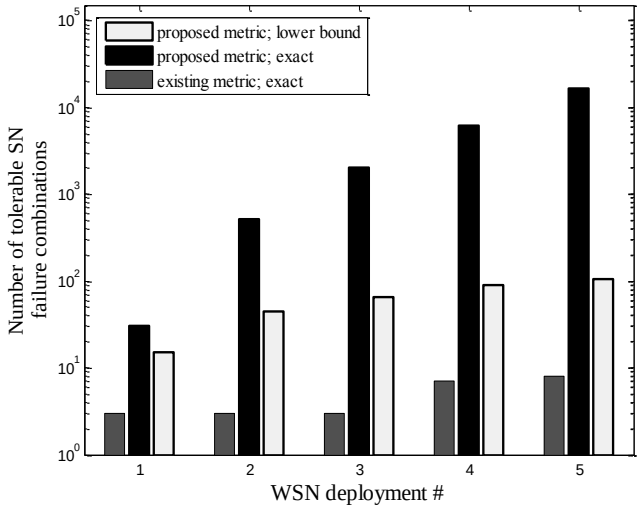


Fig. 3.7 Comparison in terms of the number of SN failure combinations contributing to the value of reliability among the existing reliability metric in [107], the proposed metric $R(\boldsymbol{S})$ and its lower bound $R_{lb}(\boldsymbol{S})$.

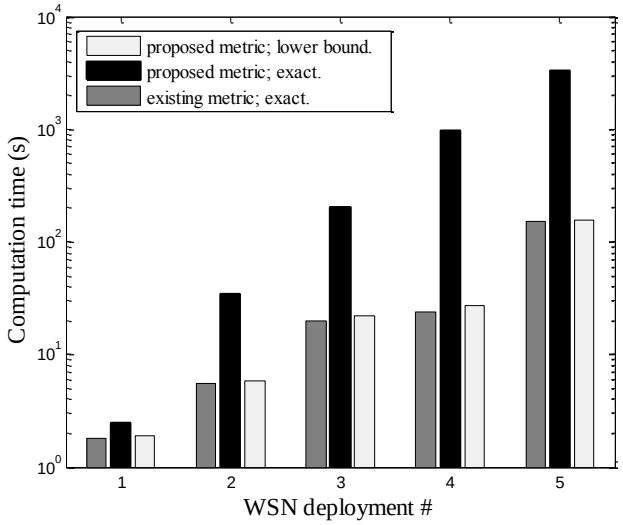


Fig. 3.8 Comparison in terms of computation time incurred by the existing reliability metric in [107], the proposed metric $R(\boldsymbol{S})$ and its lower bound $R_{lb}(\boldsymbol{S})$.

the metric proposed in [107], with the difference between them increasing with the increase in the deployment size n. This can be attributed to the use of the 3-mode SN model in the proposed metric, which raises the number of the network components subject to failure to n^2 instead of n in case of the 2-mode SN model used in [107]. This significantly increases the value of $|\boldsymbol{F}|$ (and consequently the number of structure function evaluations), which causes the gap in the computation time. However, it can be observed from Fig. 3.7 and Fig. 3.8 that $R_{lb}(\boldsymbol{S})$ offers a very good approximation for $R(\boldsymbol{S})$ at a fraction of the computation time incurred in computing $R(\boldsymbol{S})$. The computation time of the lower bound is also comparable to the computation time incurred by the metric in [107] for all five deployments in Table 3.5. This result can be attributed to the efficiency of the BFS of tolerable SN failure combinations employed in our proposed algorithm in Table 3.2. This approach enables the proposed algorithm to find the tolerable SN failure combinations of relatively higher probability before those of lower probability. Hence, the proposed algorithm finds all the tolerable SN failure combinations of probability greater than the set value of η_{lb} in a computation time comparable to that of the algorithm in [107].

3.6.3. Results and Discussion for the 3-mode, 4-par SN model

In this section we compare the proposed reliability metric using the 3-mode, 4-par SN model as expressed in (3.18) to the existing reliability metric in [107]. We also evaluate the computational efficiency of the reliability metric calculation algorithm outlined in Table 3.3 and examine the sensitivity of the computed reliability for a given deployment to the changes in the probabilities of failure of its constituent SNs.

The operational parameters of the two types of SNs used in this experiment are listed in Table 3.6. We assume that both SN types have a coverage range and a communication range of 30 and 40 meters, respectively. Similar to the parameters settings in Table 3.4, we use the reliability figures available for Texas Instrument CC2420 IEEE 802.15.4 transceiver [116] as a reference point, assuming a WSN mission time of five years and the fact that sensor hardware is the SN component most prone to failure [8]. In addition, we considered that the

premature battery failure rate for the highly durable lithium thionyl chloride batteries recently used for SNs to be very low [99].

Table 3.6 Parameters of the SNs types used in the deployments listed in Table 3.7

	FoV	r_s	r_c	λ_s	λ_t	λ_p	λ_b	**Price**($)
Type 1	90°	30 m	40 m	1.0×10^{-2}	5.0×10^{-3}	2.0×10^{-3}	1.0×10^{-3}	150
Type 2	60°	30 m	40 m	1.5×10^{-2}	5.5×10^{-3}	2.5×10^{-3}	1.5×10^{-3}	100

To evaluate computational efficiency, it is crucial to assess the effect of the deployment size and complete paths set size $|\boldsymbol{\Pi}_1|$ on the computation time of the proposed algorithm outlined in Table 3.3. Similar to the experiment in Section 3.6.2, we apply the modified iVLGA to obtain functional cost-optimized deployments. We consider five target points sets of sizes $m = 15, 20, 25, 30$ and 35. For each deployment scenario, i.e. for each value of m, we obtain five deployments of different sizes (i.e. different values of n), with the deployment of the smallest size being the most cost-optimal and the deployment with the largest size being the least cost-optimal. Each deployment fulfills the coverage and connectivity network functionality conditions in the case of no SN failures. It has a different level of SN redundancy, where the higher n, the higher the redundancy level and the larger the complete paths set $\boldsymbol{\Pi}_1$ and vice versa. Data of the resulting twenty five deployments, including the size of the deployment (n), the number of SNs of each type (n_1 and n_2) and the total deployment cost (C) are presented in Table 3.7.

To evaluate the computational efficiency of the proposed algorithm outlined in Table 3.3, we use the algorithm to evaluate the reliability of the twenty five deployments in Table 3.7. For each deployment, Table 4 shows the value of the reliability $R(\boldsymbol{S})$, the total number of possible network states $|\boldsymbol{\Pi}|$, the size of the deployment complete paths set $|\boldsymbol{\Pi_1}|$, the number of network structure function evaluations FE performed by the algorithm and the value of the ratio $FE/|\boldsymbol{\Pi}|$ in percentage points. The latter ratio is used as a measure of the computational efficiency of the proposed algorithm. This is because the most computationally expensive sub-routine in the algorithm is the evaluation of the network structure function expressed in (3.17). For each structure function evaluation, checking the two network functionality conditions, i.e. checking the network coverage of the set of target points and the connectivity to the sink, has a computational complexity of $O(n.m)$ and $O(n^3)$ respectively. Therefore, the computation time of the algorithm is mainly determined by the number of structure function evaluations FE. It should be noted that although the theoretical total number of network states is equal to 2^{4n}, the total number of *possible* network states is equal to 3^n since each deployed SN only has three possible states, namely, *on*, *relay* and *off*.

It can be readily observed that the values of $R(\boldsymbol{S})$, $|\boldsymbol{\Pi_1}|$ and increase steadily with the increase of n in each deployment scenario. This behavior is expected and is attributed to the increase in the level of SN redundancy in the deployment as n increases in each deployment scenario. An increase in the level of SN redundancy translates to an exponential increase in the number of the deployment's paths $|\boldsymbol{\Pi_1}|$and hence the number of structure function evaluations FE performed by the search algorithm to identify these paths. Naturally, as the SN redundancy level increases, the reliability $R(\boldsymbol{S})$ increases as well.

Table 3.7 Data of the obtained deployments for the case-study surveillance WSN

Deployment #		n	n_1	n_2	C (\$)	$R(\boldsymbol{S})$	$\lvert\boldsymbol{\Pi}\rvert$	$\lvert\boldsymbol{\Pi_1}\rvert$	FE	$FE/\lvert\boldsymbol{\Pi}\rvert$ (%)
Scenario 1 $m = 15$	S1-D1	9	0	9	900	0.829	3^9	4	45	0.229
	S1-D2	10	1	9	1050	0.849	3^{10}	28	164	0.278
	S1-D3	11	2	9	1200	0.870	3^{11}	196	723	0.408
	S1-D4	12	3	9	1350	0.891	3^{12}	1.37×10^3	4.16×10^3	0.783
	S1-D5	13	4	9	1500	0.912	3^{13}	9.60×10^3	3.01×10^4	1.888
Scenario 2 $m = 20$	S2-D1	16	3	13	1750	0.731	3^{16}	16	256	5.947×10^{-4}
	S2-D2	17	4	13	1900	0.748	3^{17}	112	881	6.822×10^{-4}
	S2-D3	18	4	14	2000	0.756	3^{18}	560	3.08×10^3	7.950×10^{-4}
	S2-D4	19	5	14	2150	0.774	3^{19}	3.92×10^3	1.46×10^4	1.256×10^{-3}
	S2-D5	20	6	14	2300	0.793	3^{20}	2.74×10^4	9.08×10^4	2.604×10^{-3}
Scenario 3 $m = 25$	S3-D1	21	1	20	2150	0.657	3^{21}	64	1.24×10^3	1.185×10^{-5}
	S3-D2	22	2	20	2300	0.673	3^{22}	448	4.16×10^3	1.326×10^{-5}
	S3-D3	23	3	20	2450	0.696	3^{23}	5.38×10^3	1.97×10^4	2.093×10^{-5}
	S3-D4	24	4	20	2600	0.703	3^{24}	3.23×10^4	7.49×10^4	2.652×10^{-5}
	S3-D5	25	5	20	2750	0.720	3^{25}	2.26×10^5	5.37×10^5	6.338×10^{-5}
Scenario 4 $m = 30$	S4-D1	25	8	17	2900	0.630	3^{25}	128	2.91×10^3	3.434×10^{-7}
	S4-D2	26	6	20	2900	0.612	3^{26}	192	4.51×10^3	1.774×10^{-7}
	S4-D3	27	6	21	3000	0.633	3^{27}	2.30×10^3	1.61×10^4	2.111×10^{-7}
	S4-D4	28	7	21	3150	0.649	3^{28}	1.61×10^4	6.61×10^4	2.889×10^{-7}
	S4-D5	29	8	21	3300	0.665	3^{29}	1.13×10^5	3.55×10^5	5.173×10^{-7}
Scenario 5 $m = 35$	S5-D1	28	4	24	3000	0.553	3^{28}	32	876	3.829×10^{-9}
	S5-D2	29	6	23	3200	0.555	3^{29}	48	1.34×10^3	1.952×10^{-9}
	S5-D3	30	7	23	3350	0.568	3^{30}	336	4.36×10^3	2.118×10^{-9}
	S5-D4	31	8	23	3500	0.589	3^{31}	6.38×10^3	3.04×10^4	4.922×10^{-9}
	S5-D5	32	9	23	3650	0.597	3^{32}	4.47×10^4	1.57×10^5	8.473×10^{-9}

On the other hand, the value of the ratio $FE/|\boldsymbol{\Pi}|$ decreases rapidly with the increase of the network size *across* the five tested scenarios. For example, examining the values of the ratio $FE/|\boldsymbol{\Pi}|$ for the deployments in scenarios 4 and 5 shows that there is approximately a two orders of magnitude difference in favor of the deployments in scenario 5. In all the tested deployments, the value of $FE/|\boldsymbol{\Pi}|$ does not exceed 2% and for the majority of the tested deployments is a small fraction of this value. In each scenario, the ratio $FE/|\boldsymbol{\Pi}|$ increases with the increase of the SN redundancy level due to the exponential increase of the number of the deployment's paths $|\boldsymbol{\Pi_1}|$. However, it can be observed that the ratio $|\boldsymbol{\Pi_1}|/FE$ generally increases with the increase of the level of SN redundancy in each of the five tested scenarios. For example, the value of $|\boldsymbol{\Pi_1}|/FE$ is 27% for deployment S3-D3 and 43% for S3-D4. This means that the computational efficiency of the proposed algorithm becomes more prominent with the increase of the SN redundancy level due to the efficiency of its search method for the deployment's paths.

It is instructive to examine the two deployments S4-D1 and S4-D2 which are the only exception in Table 3.7 to the trend of the increase of the reliability level with the increase of the redundancy level in each tested scenario. Although S4-D2 has more SNs than S4-D1 and a larger number of paths $|\boldsymbol{\Pi_1}|$, it is approximately 2% less reliable than S4-D1. This can be attributed to the higher ratio of more reliable SNs of type 1 to the less reliable SNs of type 2 in the S4-D1 compared to S4-D2. It can also be observed that the value of $R(\boldsymbol{S})$ decreases with the increase of the number of target points in the deployment scenarios, i.e. with the increase of m. This behavior can be explained as follows. The value of $R(\boldsymbol{S})$ depends mainly on the SN redundancy level (i.e. the value of $|\boldsymbol{\Pi_1}|$) relative to the total number of deployed SNs n (which controls the value of the probability of occurrence of the paths in $\boldsymbol{\Pi_1}$). Since the increase in n in each deployment scenario is similar, the value of $|\boldsymbol{\Pi_1}|$ for the deployments of the same order in the different scenarios (e.g. S4-D3 and S5-D3) is comparable. This means that the SN redundancy level relative to n actually decreases with the increase of deployment scenario order, i.e. with the increase of m, resulting in a steady decrease in $R(\boldsymbol{S})$.

We now compare the computed reliability values for the deployments shown in Table 3.7 using the proposed metric expressed in (3.18) to the reliability metric proposed in [107]. Since the reliability metric proposed in [107] adopts the conventional 2-mode SN model, this comparison is carried out to demonstrate the significance of modeling the SNs as three-mode (*on, relay* and *off*) devices. For a fair comparison, we use our proposed network structure function expressed in (3.17) (which defines the WSN functionality in terms of both network coverage and connectivity as opposed to network coverage only in [107]). Since the two-mode SN model assumes that a given SN is either in a fully functional (*on* state) or failed (*off* state) state, SNs cannot contribute to the network functionality as relays. Hence, the corresponding probability of the *off* state for a given SN s_i is equal to the probability that any of the four SN components fail, i.e. is equal to unity minus the probability that all of the four SN components are functioning simultaneously (i.e. $1-(1-\lambda_s^i)(1-\lambda_t^i)(1-\lambda_p^i)(1-\lambda_t^i)$).

As can be observed from Fig. 3.9 (a) – (e), the value of $R(\boldsymbol{S})$ evaluated using the 2-mode SN model is significantly smaller than that using the proposed 3-mode model for all the deployments in Table 3.7, exceeding 6% for some deployments. This behavior is expected and can be attributed to the fact that the 2-mode SN model is an unrealistic model that does not take into account the ability of an SN with a failed sensor to contribute to the functionality of the WSN in practice as a relay. Consequently, the size of the resulting paths set is drastically reduced which in turn reduces the value of $R(\boldsymbol{S})$. It should be explained that the difference between both models in $R(\boldsymbol{S})$ value for a given deployment is primarily dependent on the number of the tolerable combinations of SNs in the relay mode, i.e. the number of SNs with redundant coverage. Since this coverage redundancy is not accounted for in calculating $R(\boldsymbol{S})$ using the 2-mode SN model, the difference in $R(\boldsymbol{S})$ between the two models increases with the increase of the level of coverage redundancy. For example, the deployment S2-D5 has a higher level of coverage redundancy than S5-D5. This is reflected in their difference in $R(\boldsymbol{S})$ value between the two models, which is 5.4% for the former and 3.9% for the latter. The difference in the value of the computed reliability between the two models, although relatively limited, can adversely affect the deployment cost of a reliable WSN as will be addressed in Chapter 4, since deployment cost is the objective of the SDP while reliability is the constraint. In other word, under-evaluating the reliability of a WSN deployment can potentially lead to an increase in the deployment cost of a reliable cost-iotimal network.

In order to examine the sensitivity of $R(\boldsymbol{S})$ of a given deployment to changes in the probabilities of failure of its constituent SNs, we arbitrarily choose one of the deployments in to 0.01 for each of the four SN components, assuming the remaining components have the

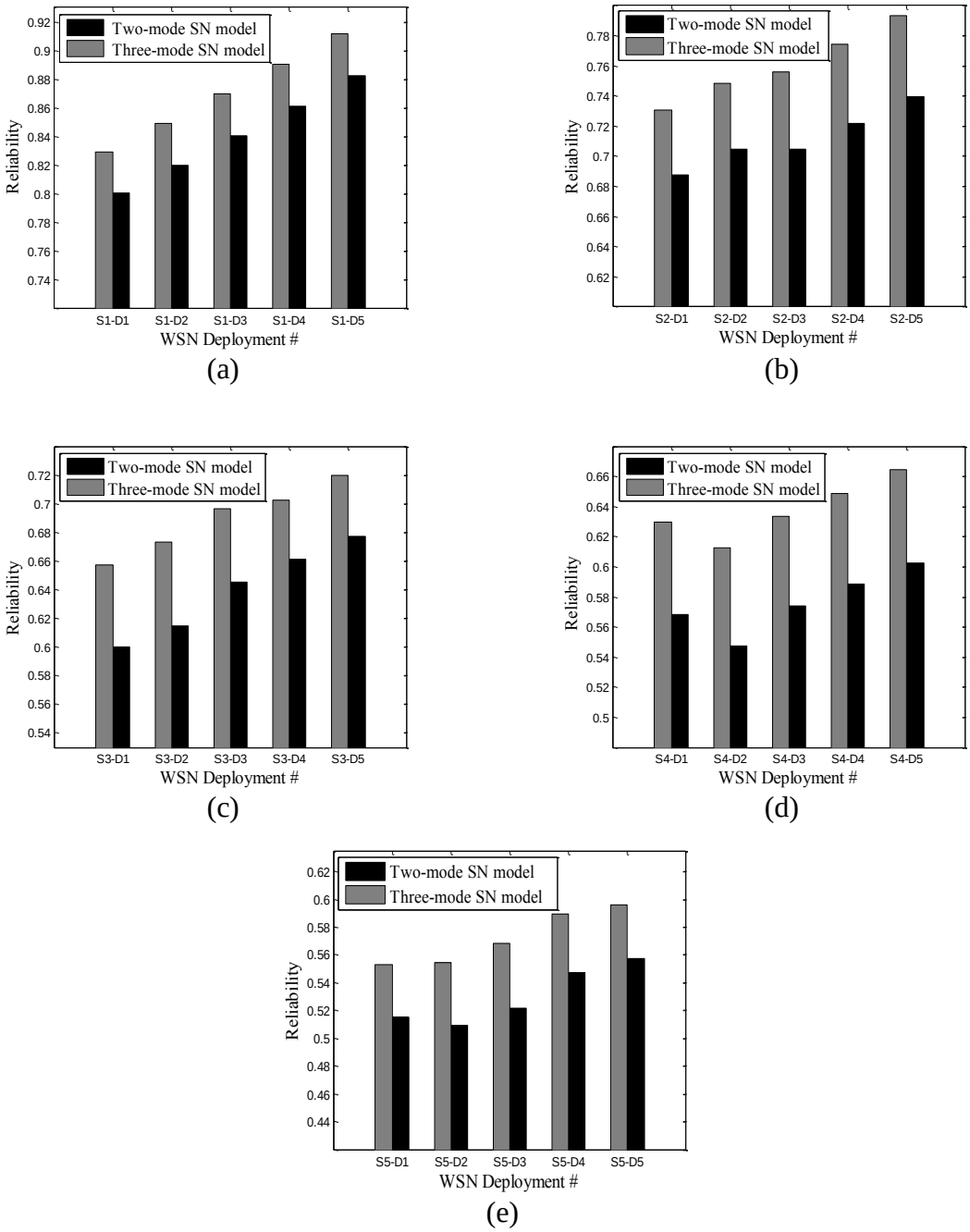


Fig. 3.9 Comparison between the reliability of WSN deployments in Table 3.7 evaluated using the proposed 3-mode and the 2-mode SN model adopted in the existing metric in [107] for the deployment scenarios 1 through 5 shown in (a) – (e).

default probabilities of failure given in Table 3.6. The results obtained are shown in Fig. 3.10. As expected, the highest value of $R(\boldsymbol{S})$ is obtained when the probabilities of failure of the four SN components are at their minimum value. Fig. 3.10 also shows that $R(\boldsymbol{S})$ is less sensitive to changes in the sensor probability of failure than to changes in the other threecomponents probabilities of failure. This can be attributed to the adopted three-mode SN model, for which the SN can contribute to the network functionality in both the *on* and *relay* modes. In the *relay* mode, the SN sensor is not functional. However, for both modes the SN battery, processor and transceiver must be functioning. Hence, the reliability of a given

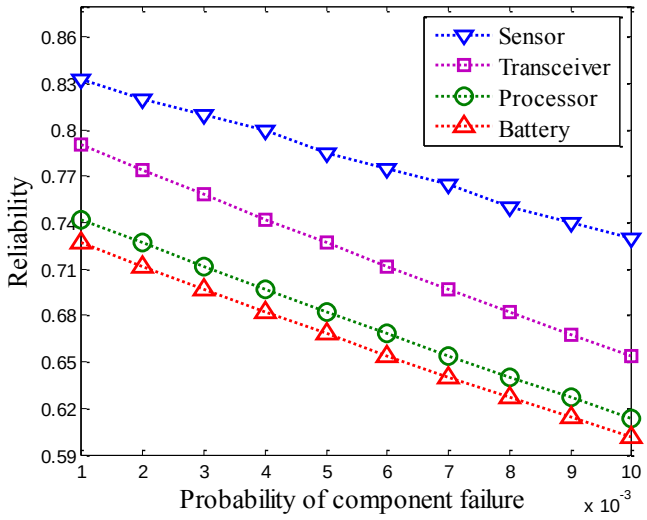


Fig. 3.10 Reliability $R(\boldsymbol{S})$ for the deployment S3-D1 in Table 3.7 at different probabilities of failure of the sensor, transceiver, processor and battery, assuming all deployed SNs are of type 1.

deployment is less affected by the change in the sensor probability of failure compared to those of the other components.

3.6.4. Comparison between the 3-mode, 2-par and 4-par SN models

In this section, we will compare the proposed reliability metric expressed in (3.11) in Section 3.5.2, to the metric expressed in (3.18) in Section 3.5.3. To carry out the comparison, we compute the reliability of the five deployments of scenario 3 in Table 3.7 using both metrics. For the 3-mode, 4-par SN model, the four failure probabilities for both types of SNs listed in Table 3.6 are used. For the 3-mode, 2-par SN model, only the sensor and transceiver failure probabilities for both types of SNs from the same table are used. The comparison is shown in Fig. 3.11. It can be observed from Fig. 3.11 that the computed reliability values using the 3-mode, 2-par SN model is higher than the values computed using the 3-mode, 4-par SN model. This result is expected since the latter model factors in the probabilities of failure of the battery and processor modules of the SNs and hence SNs have a higher probability of failure. The effect of using the more accurate 4-par model can be significant in the computed value of a given deployment reliability even for the relatively very low failure probabilities for the processor and battery modules shown in Table 3.6. For example, the relative/percentage difference between the computed reliability values from both models is approximately 9% for the deployment S3-D1.

It should be noted that the increased accuracy in calculating the reliability $R(\boldsymbol{S})$ using the 4-par model does not incur an increase in the computational cost compared to two-parameter model. This is because in both models, SNs have the same three modes of operation, which means that the total number of possible network states $|\boldsymbol{\Pi}|$ is the same for both models. The two search algorithms outlined in Tables 3.1 and 3.3 use the same search methodology for finding all the tolerable failure combinations (i.e. the complete network's paths set $\boldsymbol{\Pi_1}$) although each of them follows a different order in the search steps. Therefore, the number of structure function evaluations FE performed by the two algorithms, and hence their computational cost, is equal.

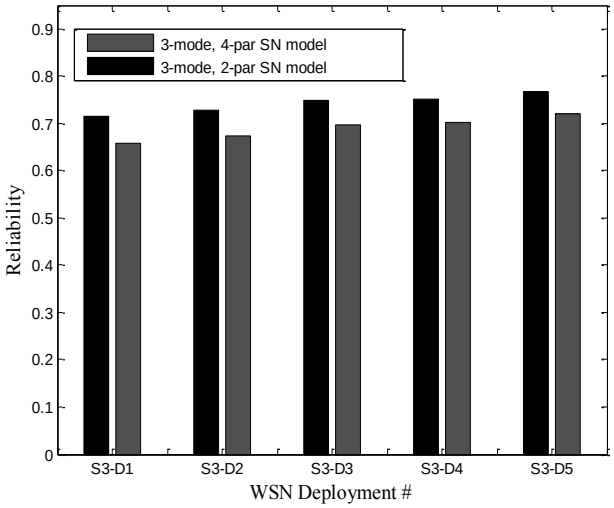


Fig. 3.11 Comparison between the reliability of WSN deployments of scenario 3 in Table 3.7 evaluated using the 3-mode, 2-par SN model and the 3-mode, 4-par SN model.

3.7. Chapter Summary

In this chapter, we identified the key SN related and non-SN related issues that affect the reliability of a WSN. We reviewed the existing studies in the literature on the reliability and fault tolerance of WSNs and highlighted their limitations. Based on the presented review, we proposed a novel reliability metric for WSNs subject to random SN failures. Compared to the existing reliability evaluation and estimation approaches, the strengths of our proposed metric can be summarized in the following points:

- Network functionality is defined in terms of both network coverage of a predefined set of target locations in the RoI and connectivity to the designated sink node.
- No specific network deployment configuration is assumed in the proposed model. We assume an arbitrary deployment configuration where each deployed SN may monitor multiple target locations in the RoI and that each target location may be monitored by multiple SNs. All SNs can communicate wirelessly with its neighbors, i.e. no specific communication hierarchy is imposed.
- The WSN can be heterogeneous; it can consist of more than one type of SN, where each type is characterized by a different coverage profile and a set of capabilities.
- A more accurate SN model is adopted in the derivation of the proposed metric where an SN has three modes of operation instead of the two-mode model used in the previous studies.
- Each SN type can be characterized by two or four different probabilities of constituent module failure during the mission time of the network instead of a single SN probability of failure, as it is the case in the previous studies.
- A search algorithm is developed to calculate the proposed reliability metric in a computationally efficient manner for each SN model.

We applied the proposed metrics and search algorithms experimentally to several deployments of a surveillance WSN under different operational conditions. Results demonstrated the computational efficiency of the developed search algorithms. Moreover, the significance of adopting the proposed 3-mode SN model on the evaluated value of WSN reliability as opposed to the conventional simplistic 2-mode SN model adopted in existing

studies can be observed in the results. Using the 2-mode SN model can significantly underestimate the reliability of a WSN deployment since it does not account for the *relay* SN state as in the 3-mode SN model.

Chapter 4

Reliable Cost-Optimal Wireless Sensor Network Deployment

4.1. Introduction

Some of the important IoT applications in which WSNs play a pivotal role place stringent reliability requirements on the WSN. Reliability of the WSN is therefore considered one of the most essential design attributes. In such applications, the failure of the network to carry out its required tasks can have serious effects and hence cannot be tolerated.

As previously discussed, a reliable WSN provides a connected cover of the targeted RoI throughout its mission time. However, the deployment of reliable WSNs is a challenging problem due to the random failures of the SNs. Hence, to guarantee the reliable operation of a WSN during its intended mission time, the presence of redundant SNs in the network becomes essential. However, for many applications for which SNs are equipped with expensive hardware, minimizing the total deployment cost remains a primary concern. Therefore, the level of SN redundancy in the WSN must be carefully quantified, such that the network meets the minimum reliability requirements imposed by the application while avoiding an unnecessary increase in the network deployment cost.

In this chapter, we formally define the problem of deploying a WSN that meets a specified minimum level of reliability defined over a given mission time in such a way that result in the minimum network deployment cost. In Chapter 1, we coined this problem the MCRC-SDP. We formulate the MCRC-SDP as a combinatorial optimization problem and prove that it is NP-Complete. The performance of GAs and ACO has proven to be promising in solving complex combinatorial NP-Complete optimization problems [117] - [121] and in solving the MCC-SDP as demonstrated in Chapter 2. Therefore, we propose a GA-based and an ACO-based method to solve the defined MCRC-SDP. Both methods are coupled with a Local Search (LS) procdeure to improve the method's search capability and increase its speed of convergence. To measure the reliability of the network, we adopt the reliability metric proposed in Chapter 3. To benchmark the performance of both methods in solving the problem at hand in terms of the quality of the obtained solutions, we also present a GH which is designed to solve the MCRC-SDP. Finally, we present extensive experimental results which we use to compare the two proposed methods, in terms of both the quality of the obtained solutions and the computational cost. We then discuss their strengths and limitations.

4.2. Minimum-Cost Reliability-Constrained SDP

In this section, we formally define the MCRC-SDP as a combinatorial optimization problem. We start by defining the WSN model in Section 4.2.1. We then mathematically formulate the MCRC-SDP in Section 4.2.2. Finally we prove that the MCRC-SDP is NP-Complete in Section 4.2.3.

4.2.1. WSN Model

We adopt here the same WSN model used in formulating the WSN reliability metric presented in Section 3.5 in Chapter 3. We assume that the RoI is modeled as a two-dimensional area in which there is a finite set of locations that require some form of monitoring (e.g. motion, image...etc.) using static SNs. These locations are called *target points* and they represent the vital locations or assets that require monitoring in the RoI. We denote the set of target points $\boldsymbol{T} = \{t_1, t_2, \dots, t_{|\boldsymbol{T}|}\}$. We assume that there is a finite set of possible deployment locations for SNs, which we call *deployment points*, at which SNs may be deployed. This assumption is valid for most critical WSN applications, where the topology or layout of the targeted RoI is known prior to the WSN deployment. Hence, careful examination of that RoI yields a finite set of feasible possible deployment locations, i.e. deployment points. We denote the set of deployment points $\boldsymbol{D} = \{d_1, d_2, \dots, d_{|\boldsymbol{D}|}\}$.

All SNs available for deployment are assumed to be able to communicate wirelessly and have the same fixed communication range denoted by r_c. Sensed data acquired by the deployed SNs are relayed to a sink node with an arbitrary fixed position in the RoI denoted by d_0.

4.2.2. Problem Formulation

We address the problem of deploying a WSN that meets a specified minimum level of reliability, denoted by R_{min}, defined over a given mission time at the minimum network deployment cost. The reliability requirement of the MCRC-SDP implicitly includes *three* sub-requirements. The first two sub-requirements are the fulfillment of the coverage and connectivity functionality aspects according to the WSN model and the network functionality definition presented in Section 3.5.1 in Chapter 3. The third sub-requirement is that the WSN must possess a certain level of robustness against the random failures of its constituent SNs such that the network can fulfill the coverage and connectivity functionality conditions *throughout* the network mission time despite random SN failures. This robustness, in turn, requires introducing a certain level of SN redundancy in the network deployment. However, the introduction of redundant SNs in a WSN can significantly increase the energy consumption of the network, the demand on its limited bandwidth and its level of internal interference [100] under the assumption that all the deployed SNs are activated at the same time. This introduces non-SN related issues (e.g. excessive packet collisions in WSNs adopting contention-based medium access control) which negatively affect the reliability of the message delivery in the network, thus defeating the purpose of introducing the redundancy in the first place. Therefore, SN activity planning is required to increase the robustness of the WSN against SN failures without introducing significant degradation in its performance.

As such, we can restate the MCRC-SDP to be the problem of finding a number of non-overlapping *minimal* connected covers of the targeted RoI such that the combined reliability level of these minimal connected covers would meet or exceed the specified minimum level of reliability R_{min} and the total number of deployed SNs (i.e. the deployment cost) is minimized. A minimal connected cover is defined as a connected cover which contains no completely redundant SNs. These minimal connected covers are activated in an orthogonal manner as follows: a single minimal connected cover is activated at any given point in time during T_m while the SNs belonging to the remaining connected covers are put in sleep mode. Since there are no completely redundant SNs in a minimal connected cover, energy consumption, bandwidth usage and internal interference are kept at a minimum. This activated minimal connected cover remains active until its functionality is compromised due

to the expected random failures of its constituent SNs. At that point, the remaining functional SNs belonging to this minimal connected cover are put in sleep mode and another minimal connected cover is activated. This procedure is continued until either the mission time of the network T_m elapses or there are no remaining deployed minimal connected covers of uncompromised functionality. The first event means the WSN deployment remained functional throughout T_m while the second event means that the WSN has failed. According to the statement of the problem, the probability of the first event is equal to R_{min} and that of the second event is equal to $1 - R_{min}$.

Let $\boldsymbol{\mathcal{S}} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \dots, \boldsymbol{S}_N\}$ be the superset of N non-overlapping connected covers in a given WSN deployment. For simplicity, we assume here that the WSN is homogeneous, i.e. composed of the same type of SNs. The MCRC-SDP can then be formulated as follows:

$$min\left\{|\boldsymbol{\mathcal{S}}| = \sum_{k=1}^{N} |\boldsymbol{S}_k|\right\}, \tag{4.1}$$

subject to:

$$\boldsymbol{S}_k \subseteq \boldsymbol{D} \quad \forall\, k = 1, \dots N, N \leq N_{UB}\,, \tag{4.2}$$

$$\boldsymbol{S}_k \cap \boldsymbol{S}_{k'} = \phi, \qquad \forall\, k, k' = 1, \dots, N, k \neq k'\,, \tag{4.3}$$

$$R(\boldsymbol{\mathcal{S}}) = R(\boldsymbol{S}_1, \boldsymbol{S}_2, \dots, \boldsymbol{S}_N) = 1 - \prod_{k=1}^{N} \big(1 - R(\boldsymbol{S}_k)\big) \geq R_{min}\,, \tag{4.4}$$

$$\Phi(\boldsymbol{S}_k) = 0 \quad \forall\, k = 1, \dots, N. \tag{4.5}$$

Equation (4.1) is the objective function of the MCRC-SDP, which is the minimization of the total number of the deployment points (i.e. deployed SNs) belonging to all N disjoint minimal connected covers, i.e. $|\boldsymbol{\mathcal{S}}|$. This is equivalent to the minimization of the network deployment cost. Equations (4.2) - (4.5) are the constraints of the problem. In (4.2), all the connected covers are constrained to be subsets of the deployment points set $\boldsymbol{D}$ in the targeted RoI. Equation (4.2) also sets the number of connected covers in the solution $\boldsymbol{\mathcal{S}}$ denoted by N to be less than or equal to N_{UB}, which is defined as the the upper bound on the number of connected covers for a given MCRC-SDP instance, i.e. for a given $\{\boldsymbol{T}, \boldsymbol{D}\}$ tuple. The process of estimating this upper bound is detailed in Section 4.2.3. Equation (4.3) expresses the condition that the N minimal connected covers constituting $\boldsymbol{\mathcal{S}}$ must be disjoint, i.e. have no deployment points in common. Equation (4.4) expresses the reliability constraint of the problem. In (4.4), the total reliability of the WSN deployment, i.e. of $\boldsymbol{\mathcal{S}}$, is calculated in terms of the reliability of the N connected covers assuming they are activated orthogonally. We will measure the reliability of the connected covers using the reliability metric presented in Section 3.5.3. Finally, (4.5) constrains each of the N connected covers in $\boldsymbol{\mathcal{S}}$ to be a minimal connected cover, where $\Phi(\boldsymbol{S}_k)$ is a binary function that returns 0 if the connected cover $\boldsymbol{S}_k$ is a minimal connected cover and 1 otherwise. Note that if $\boldsymbol{S}_k$ is a minimal connected cover, there would be no completely redundant SNs in $\boldsymbol{S}_k$. This means that there would be no combinations of SNs in the *off*-mode that would correspond to a unity network structure function ($f(\boldsymbol{\pi}) = 1$), i.e. $\boldsymbol{F}_o = \{\phi\}$. However, this does not mean that there would not be *coverage* redundant SNs in a minimal connected cover $\boldsymbol{S}_k$. That is, it is possible for $\boldsymbol{S}_k$ to have one or more combinations of SNs in the *relay*-mode that would correspond to a unity network structure function which means that $\boldsymbol{F}_r \neq \{\phi\}$.

4.2.3. Estimation of the Upper-Bound of the Number of Connected Covers

To estimate the upper bound of the number of connected covers denoted by N_{UB} in a given MCRC-SDP instance represented by a given $\{\boldsymbol{T},\boldsymbol{D}\}$ tuple, we make use of the fact that the upper bound for the number of connected covers cannot exceed the upper bound for the number of *covers*. A cover in a given MCRC-SDP instance is a subset of $\boldsymbol{D}$ which meet the coverage constraint only. Thus the upper bound of the covers can be used as the upper bound of the number of connected covers N_{UB}. Although finding the maximum number of covers for a given MCRC-SDP instance is an NP-complete problem, we can estimate the upper bound of the number covers with the following method [121]. Assume that all the deployment points in $\boldsymbol{D}$ have SNs deployed on them. Then, locate the least covered target point in $\boldsymbol{T}$, i.e. the target point(s) with the smallest number of SNs covering it. We will call this target point a *critical* target point(s). The number of SNs covering the critical target point(s) represents the upper bound on the number of covers and hence connected covers denoted by N_{UB}. This is because a cover of the RoI cannot provide full coverage of $\boldsymbol{T}$ without providing coverage of the critical target point(s). Hence, the maximum number of covers cannot exceed the number of SNs covering the critical target point(s). To illustrate this, Fig. 4.1 shows a problem instance where $\boldsymbol{T} = \{t_1, t_2, t_3\}$ and $\boldsymbol{D} = \{d_1, d_2, d_3, d_4, d_5, d_6\}$. We will assume here that the SNs available for deployment have a disk-coverage profile. As can be observed from the figure, three SNs provide coverage for target points t_1 and t_3. For t_1 the SNs are located on deployment points $\{d_1, d_2, d_6\}$ while for t_3 they are located on $\{d_2, d_3, d_5\}$. Target point t_2, on the other hand, is covered by only two SNs deployed on deployment points $\{d_3, d_4\}$. As such, t_2 is the critical target point and the upper bound of the number of connected covers N_{UB} for this problem instance is equal to 2. Although the authors in [121] did not comment on the tightness of the above method in estimating N_{UB}, it can be deduced that the estimate N_{UB} can actually be equal to the exact value in the case where $r_c \gg r_s$ such that all SNs are within as single hop of the sink node. In all other cases, the difference between N_{UB} and the actual value of the maximum number of connected covers depends on the relative spatial positions among the deployment points, target points and the sink node.

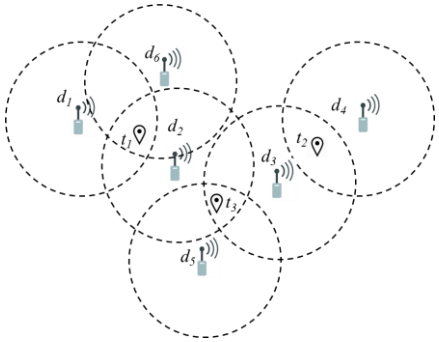


Fig. 4.1 A RoI containing three target points $\boldsymbol{T} = \{t_1, t_2, t_3\}$ and six deployment points $\boldsymbol{D} = \{d_1, d_2, d_3, d_4, d_5, d_6\}$. Target point t_2 is the critical target point and the upper bound for connected covers is $N_{UB} = 2$.

4.2.4. Proof that MCRC-SDP is NP-Complete

To prove that the MCRC-SDP expressed in (4.1) - (4.5) is NP-complete, we start by considering the decision problem that corresponds to the MCRC-SDP. We will call this

decision problem the Reliability Constrained SN Deployment Problem (RC-SDP). The RC-SDP can be expressed as follows.

RC-SDP: given $\boldsymbol{D}, \boldsymbol{T}, N_{UB}(\, N_{UB} \in \mathrm{Z}^{+}), R_{min}$ $(R_{min} \in [0,1])$, does a superset $\boldsymbol{\mathcal{S}} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \dots, \boldsymbol{S}_N\} \subseteq \boldsymbol{D}$ exist, such that the following conditions are true?

1. $\boldsymbol{S}_k \subseteq \boldsymbol{D} \quad \forall\, k = 1, \dots N, N \leq N_{UB}$
2. $\boldsymbol{S}_k \cap \boldsymbol{S}_{k'} = \varphi, \qquad \forall\, k, k' = 1, \dots, N, k \neq k'$;
3. $R(\boldsymbol{\mathcal{S}}) = 1 - \prod_{k=1}^{N}\big(1 - R(\boldsymbol{S}_k)\big) \geq R_{min}$;
4. $\Phi(\, \boldsymbol{S}_k\,) = 0 \quad \forall\, k = 1, \dots, N.$

Theorem 1: RC-SDP is NP

Proof: To prove that RC-SDP is NP, we need to prove that for any given superset $\boldsymbol{\mathcal{S}} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \dots, \boldsymbol{S}_N\} \subseteq \boldsymbol{D}$, we can check if $\boldsymbol{\mathcal{S}}$ fulfills the problem's conditions in polynomial time. The computational complexity of checking each of the problem's conditions is given, in order, as follows:

1. $O(|\boldsymbol{D}|)$, where $\sum_{k=1}^{N} |\, \boldsymbol{S}_k| = |\boldsymbol{\mathcal{S}}| \leq |\boldsymbol{D}|$;
2. $O(max_k\{|\, \boldsymbol{S}_k|^2\})$, $k = 1, \dots, N$;
3. $O(max_k\{|\, \boldsymbol{S}_k|^4\})$;
4. $O(max_k\{|\, \boldsymbol{S}_k|^4\})$.

Checking the third and fourth conditions has the same computational complexity since checking whether a connected cover $\boldsymbol{S}_k$ is minimal or not comes automatically through the process of calculating its reliability $R(\, \boldsymbol{S}_k)$. The computational complexity of calculating $R(\, \boldsymbol{S}_k)$ is dictated by the complexity of the structure function evaluation, which is the most computationally expensive routine in the reliability calculation algorithm presented in Section 3.5.3.3 in Chapter 3. To evaluate the network structure function $f(\boldsymbol{\pi})$ at a given network state $\boldsymbol{\pi}$, the two network functionality conditions needs to be checked. Evaluating the network coverage of the set of target points $\boldsymbol{T}$ and that of the connectivity to the sink at a given network state have a computational complexity of $(|\, \boldsymbol{S}_k| \, * |\boldsymbol{T}|)$ and $(|\, \boldsymbol{S}_k|^3)$, respectively. This is valid under the assumption that in the presence of coverage redundancy, only a fixed number of maximum coverage functionality checks is allowed when evaluating the reliability in the third condition. This gives an overall computational complexity of $O(arg\, max_k\{|\, \boldsymbol{S}_k|^3\})$, where $0 < |\, \boldsymbol{S}_k\, | \leq |\boldsymbol{D}|$.

Therefore, RC-SDP is NP ■

Theorem 2: RC-SDP is **NP**-hard

Proof: We use the method of restriction to prove that RC-SDP is NP-hard. For any given problem instance, i.e. for a given $\{\boldsymbol{T}, \boldsymbol{D}\}$ tuple, let $R_{min} = \epsilon \ll 1$. This means that we are looking for a deployment that consists of a single minimal connected cover deployment and that any non-zero value of reliability is acceptable. This restriction converts the RC-SDP to the problem of deciding whether there exists a single minimal connected cover $\boldsymbol{S} \subseteq \boldsymbol{D}$ of size/cardinality $|\boldsymbol{S}| \leq |\boldsymbol{D}|$ that provides full coverage of $\boldsymbol{T}$ and is connected to the given sink node. This latter problem has been proved NP-complete in [122].

Therefore, RC-SDP is NP-hard ■

From theorems 1, 2 → RC-SDP is NP-complete ■

4.3. Proposed Optimization Algorithms for Solving the RCSDP

Since the MSCRC-SDP is NP-Complete, solving instances of the MCRC-SDP of practical scale using exact optimization methods (such as Integer Linear Programming (ILP), the Branch and Bound method (B&B) and the Branch and Cut method (B&C)) is not computationally feasible. This is due to the fact that the calculation time for these

optimization methods increases exponentially with the problem size [123]. For example, an efficient implementation of the B&B for the famous Travelling Salesman Problem (TSP) has a computational time complexity $O(n^2 * 2^n)$[124] where n is the number of cities in the problem. On the other hand, heuristic or stochastic optimization methods offer a viable alternative to the exact optimization methods in solving complex large-scale optimization problems. Generally speaking, they are capable of reaching good solutions for these problems in a relatively short amount of calculation time. Although these methods do not guarantee reaching the global optimum solution of the problem, they can often lead to near-optimal solutions that are slightly worse than the global optimum if they are well designed. Moreover, a generic form of a stochastic optimization method can be tuned according to the special characteristics of an optimization problem, enabling the method to reach even better solutions, possibly even the global optimum solution [123].

GAs and ACO algorithms are among the most widely used stochastic optimization methods. Their performance has proven to be promising in solving complex combinatorial NP-Complete optimization problems [125] - [131]. In the context of WSN deployment, their effective performance in solving the MCC-SDP has been demonstrated in Section 2.5.2 in Chapter 2. It should be noted that other first-order derivative iterative optimization methods such as gradient descent can be applied to the MCRC-SDP. However, the complex nature of the reliability constraint expressed in (4.4) makes it difficult to use. In this section, we present two reliable cost-optimal deployment algorithms using both methods and analyzing their obtained results to evaluate their performance according to the two aforementioned metrics: the quality of the obtained solutions and the computational speed/cost. Coupling both methods with an LS procedure has been reported to increase the method's speed of convergence and enhances its search capability [117] - [121]. We therefore have each of the proposed algorithms apply an LS procedure suitable for its design. For each of the proposed algorithms, we will discuss the design of the fundamental building blocks, the LS procedure and the termination conditions of the algorithm.

4.3.1. Proposed Memetic Algorithm

In this Section, we present the proposed Memetic Algorithm (MA) for solving the MCRC-SDP problem expressed in (4.1) - (4.5). The term Memetic Algorithm is used in literature to refer to a combination of an evolutionary-based algorithm, such as a GA, with a LS procedure customized to the problem at hand, also known as a *meme* [118]. This combination is also referred to as a Hybrid GA (HGA). In the following sub-sections, we discuss the different building blocks of the proposed MA, namely, the chromosome-encoding scheme, the fitness function, the chromosome selection schemes, the variation operators, the applied LS procedure and finally, the termination conditions of the algorithm.

4.3.1.1. Chromosome Encoding Scheme

We select to use an integer-encoding scheme for the MA chromosome encoding. Each chromosome is composed of $|\boldsymbol{D}|$ genes, where each gene represents one of the deployment points in the set $\boldsymbol{D}$ with an ordered one-to-one correspondence. The value given to each gene varies between 0 and N_{UB}. The value of the gene indicates whether an SN is deployed at the corresponding deployment point (if it takes a non-zero value between 1 and N_{UB}) or not (if it is null). If a given gene takes a non-zero integer value, then this value indicates the index of the SN set which the corresponding deployment point (and hence the actual SN deployed on it) belongs to. Each SN set represented in a chromosome is a potential connected cover of the targeted RoI, i.e. the set of target points $\boldsymbol{T}$, if that SN set fulfills the coverage and connectivity conditions defined in Section 3.5.1 in Chapter 3. Therefore, the maximum number of SN sets

represented in any given chromosome is N_{UB} and hence the upper bound for the number of connected covers is equal to N_{UB} as well. The ordered one-to-one correspondence between the genes of the chromosome and the deployment points in $\boldsymbol{D}$ ensures that all the represented connected covers in a given chromosome are disjoint and hence the MCRC-SDP constraints expressed in (4.2) and (4.3) are always satisfied.

For example, consider the problem instance illustrated in Fig. 4.1 where $\boldsymbol{T} = \{t_1, t_2, t_3\}$, $\boldsymbol{D} = \{d_1, d_2, d_3, d_4, d_5, d_6\}$ and $N_{UB} = 2$. Fig. 4.2 shows a possible chromosome encoding using the proposed scheme and how it is decoded to a possible solution to the problem at hand. Since $|\boldsymbol{D}| = 6$ in this example, the number of genes in the chromosome is 6. The genes in the chromosome take integer values between 0 and 2 since $N_{UB} = 2$ as explained in Section 4.2.3. The chromosome in Fig. 4.2 corresponds to two SN sets, the first one is composed of deployment points d_1, d_3 and d_4 and the second one is composed of deployment points d_2 and d_6. For simplicity, we will assume that in this example that all six deployment points are within r_c of the sink node, i.e. any cover of $\boldsymbol{T} = \{t_1, t_2, t_3\}$ is automatically a connected cover. It is clear that the first SN set is a connected cover while the second one is not. Therefore the chromosome in Figure 2 corresponds to one connected cover $\boldsymbol{S}_1 = \{d_1, d_3, d_4\}$. It is important to point out that the problem's reliability and redundancy constraints expressed in (4.4) and (4.5) are not automatically met in each chromosome, since it is possible that a given chromosome does not represent enough connected covers to meet the specified minimum reliability R_{min} or that one or more of the connected covers is not a minimal connected cover. For example, the connected cover represented by the chromosome in Fig. 4.2 is not a minimal connected cover since d_4 is completely redundant. Hence, only a part of the genotypic space corresponds to feasible solutions to the MCRC-SDP. Therefore, these constraints must be incorporated in the fitness function of the proposed MA as discussed in the next sub-section.

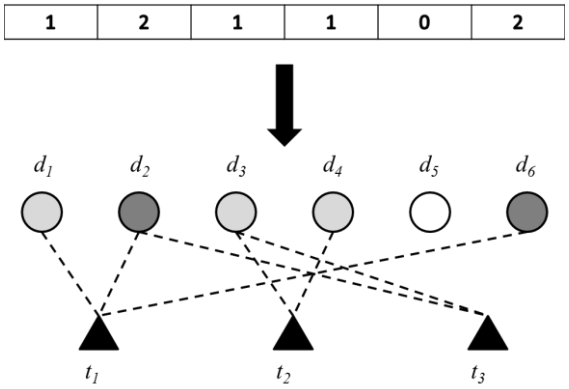


Fig. 4.2 Chromosome decoding for the proposed MA: the chromosome corresponds to two SN sets, the first set includes d_1, d_3 and d_4 and the second set includes d_2 and d_6. The value of the fifth gene is null, therefore d_5 is not assigned to a SN set. Since only the first SN set covers $\boldsymbol{T} = \{t_1, t_2, t_3\}$, the chromosome corresponds to a single connected cover $\boldsymbol{S}_1 = \{d_1, d_3, d_4\}$.

4.3.1.2. Fitness Function

Since the objective function of the MCRC-SDP expressed in (4.1) is minimizing the total deployment cost of the WSN, the fitness of any given chromosome must be inversely proportional to the total number of deployment points (i.e. deployed SNs) belonging to all the connected covers represented in the chromosome. However, since only a part of the genotypic space corresponds to feasible solutions that satisfy the minimum reliability and redundancy constraints expressed in (4.4) and (4.5), the fitness function must also incorporate

these two constraints by the means of *penalty* terms. This is a common practice in using GAs to solve constrained optimization problems [130]. The penalty terms work by dampening the fitness of unfeasible solutions that do not meet one or more of the constraints in order to direct the search away from the neighborhoods of these solutions in the genotypic space.

Let $c(N)$ be a chromosome representing N disjoint connected covers, i.e. a chromosome that corresponds to a WSN deployment $\boldsymbol{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \dots, \boldsymbol{S}_N\}$, where $1 \leq N \leq N_{UB}$. The fitness value given to $c(N)$, denoted by $\mathcal{F}(c(N))$, is calculated using the following fitness function:

$$\mathcal{F}(c(N)) = -\left(\sum_{k=1}^{N}|\boldsymbol{S}_k| + \omega_1 \sum_{k=1}^{N} \Phi(\boldsymbol{S}_k) + \omega_2\left[R_{min} - \left(1 - \prod_{k=1}^{N}(1 - R(\boldsymbol{S}_k))\right)\right]^{+}\right), \quad (4.6)$$

where the first term of the function represents the total deployment cost which is equal to the total number of the deployment points belonging to the N disjoint connected covers (i.e. $|\boldsymbol{S}|$) represented in $c(N)$. The second term penalizes the fitness of the chromosome for every non-minimal connected cover in $\boldsymbol{S}$, for which $\Phi(\boldsymbol{S}_k)$ is equal to unity, by a value equal to the constant weight, ω_1. The third term penalizes the fitness of the chromosome if the collective reliability of the disjoint connected covers represented in the chromosome is less than the specified minimum reliability level, R_{min}. The value of the penalty is equal to the difference between the two reliability levels multiplied by the constant weight, ω_2. The constant weights ω_1 and ω_2 are set such that fitness values assigned to the chromosomes follow the following scale. All the chromosomes which meet both the reliability and redundancy constraints expressed in (4.4) and (4.5) respectively have higher fitness values than all the chromosomes that fail to meet either constraint. On the other hand, all the chromosomes which meet the reliability constraint have higher fitness values than all the chromosomes which do not meet the reliability constraint, whether they meet the redundancy constraint or not. This approach in assigning fitness will direct the MA search to the most promising regions in the genotypic space which correspond to high quality feasible solutions to the MCRC-SDP. Since the maximum value for $\sum_{k=1}^{N}|\boldsymbol{S}_k|$ (i.e. the deployment cost) is $|\boldsymbol{D}|$ and the upper bound for the number of connected covers is N_{UB}, it is easy to verify that setting ω_1 to the value $|\boldsymbol{D}| + 1$ and ω_2 to $(|\boldsymbol{D}| + 1)(N_{UB} + 1) * 10^{n}$ will fulfill the required fitness scaling described above, where n is the number of significant decimal places in both R_{min} and $R(\boldsymbol{S}_k)$.

4.3.1.3. Variation Operators

The traditional variation operators, i.e. crossover and mutation operators, for integer encoded GAs are applicable for the proposed chromosome encoding scheme. In the proposed MA, we adopt a simple single-point crossover and creep mutation [130]. For each pair of parent chromosomes, the single-point crossover operator chooses the crossover point at random in the interval $[1, |\boldsymbol{D}|]$ and creates two offspring by exchanging parts of the parent chromosomes. The creep mutation simply changes the value of a gene in an offspring chromosome to a value in the interval $[0, N_{UB}]$.

We adopt an additional variation operator to the standard genetic operators called *scattering* [131], which is applied to the offspring population directly after crossover and mutation. This operator is used to help the proposed MA in avoiding regions in the genotypic space that correspond to infeasible solutions to the MCRC-SDP. To explain how the operator works, we define the term *critical deployment points*, which refer to the deployment points which have the critical target point in the RoI (as defined in Section 4.2.3) within their coverage region. For example, consider the problem instance illustrated in Fig. 4.1, where the critical target point is t_2 and hence the critical deployment points are d_3 and d_4. The scattering operator distributes the critical deployment points on the different possible SN sets

in each chromosome. Since a connected cover must include at least one critical deployment point to provide full coverage, the scattering operator increases the chance of the creation of connected covers in each chromosome. This, in turn, increases the chance of the chromosomes translating to feasible solutions to a given instance of the MCRC-SDP.

The scattering operator works as follows. It checks the genes which correspond to the critical deployment points for the given MCRC-SDP instance, i.e. for a given $\{\boldsymbol{T}, \boldsymbol{D}\}$ tuple. If it finds that two or more of these genes are given same value, it changes the repeated genes to other values such that each of these genes is given a unique integer value in the interval $[1, N_{UB}]$. For example, applying the scattering operator on the chromosome illustrated in Fig. 4.2 will change the value of the fourth gene, which represents the critical deployment point d_4 from 1 to 2 , since the other critical deployment point d_3 is already set to 1. Accordingly, the altered chromosome will now represent two connected covers, namely $\boldsymbol{S}_1 = \{d_1, d_3\}$ and $\boldsymbol{S}_2 = \{d_2, d_4, d_6\}$ as shown in Fig. 4.3.

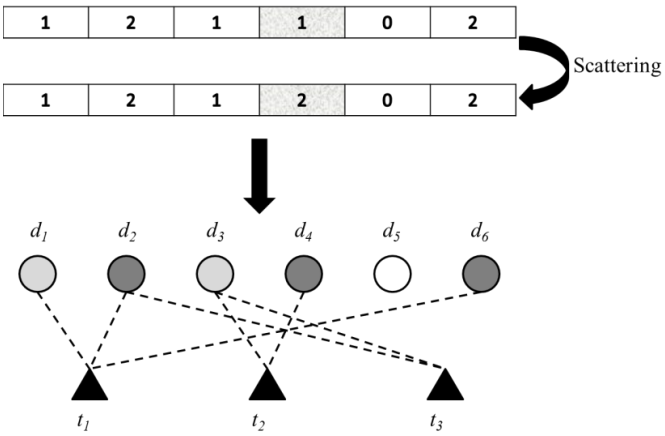


Fig. 4.3 The scattering operator in the proposed MA: the value of the fourth gene in the chromosome in Fig. 4.2 is changed from 1 to 2. The altered chromosome now represents two connected cover $\boldsymbol{S}_1 = \{d_1, d_3\}$ and $\boldsymbol{S}_2 = \{d_2, d_4, d_6\}$.

4.3.1.4. Chromosome Selection Methods

Two types of chromosome selection methods are required in the design of the proposed MA. The first one is the parent selection method, which dictates how the parent chromosomes in a current population are chosen to undergo crossover. In our proposed MA, we adopt the widely-known Roulette Wheel parent selection method [130]. Assuming the number of chromosomes in a population is μ, the Roulette Wheel method is applied to the entire population to select $\mu/2$ pairs of parents and hence μ offspring chromosomes are produced after the crossover, mutation and scattering operators are applied. The second selection method is the survivor selection method, which determines which chromosomes in the aggregated pool of parents and offspring populations of size 2μ will survive to the next generation/iteration of the algorithm. We adopt a fitness-based survivor selection which selects the μ chromosomes with the highest fitness from that pool to constitute the next generation/iteration. This selection method is also known as the $\mu + \lambda$ selection scheme [130].

4.3.1.5. Local Search Procedure

As explained earlier, the proposed MA is composed of a GA coupled with a LS procedure that helps the GA fine tune its search for high quality solutions to the problem at

hand in the promising regions of the genotypic space. In each iteration of the proposed MA after the selection of the surviving chromosomes for the following generation/iteration, we apply an LS procedure to a fraction of the chromosome population denoted by P_{LS}. The chromosomes which undergo the LS in each generation/iteration are selected at random. To strike the best balance between the global and the local search and to avoid the premature convergence of the proposed MA, we adopt a gradual increase in P_{LS} with the number of the performed generations/iterations of the algorithm [132]. Fig. 4.4 shows the adopted scheduling scheme for the application of the LS procedure in our proposed MA, where n_{LS} denotes the number of iterations at which the entire chromosome population undergoes the LS procedure, i.e. $P_{LS} = 1$.

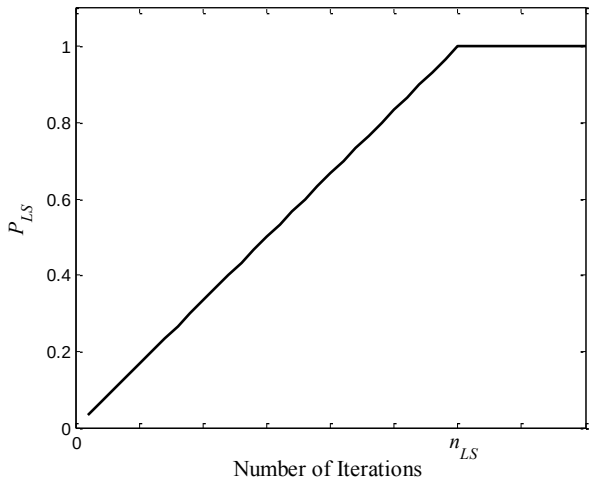


Fig. 4.4 The LS procedure scheduling scheme in the proposed MA: the fraction of the chromosome population undergoing the LS procedure P_{LS} versus the number of performed iterations of the algorithms.

Table 4.1 shows the pseudo-code of the LS procedure in the proposed MA. The operation of the LS procedure can be described as follows. The LS is applied on the input chromosome $c(N)$ that has a fitness of $\mathcal{F}(c(N))$, which corresponds in the phenotypic space to the solution $\boldsymbol{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \dots, \boldsymbol{S}_N\}$ with a combined reliability $R(\boldsymbol{S})$. The first step of the LS procedure (line 2) is to initialize the resulting chromosome of the LS and its fitness, denoted by $c_{LS}(N)$ and $\mathcal{F}\big(c_{LS}(N)\big)$ respectively, to the input chromosome and its fitness. In this LS procedure, we adopt a Lamarckian approach, in which both the fitness and the genotypic representation of the solution, i.e. the chromosome, are changed by the LS procedure [132]. In the second step (lines $4-10$), the LS checks if any of the connected covers in $\boldsymbol{S}$ violates the redundancy constraint in (4.5), i.e. $\Phi(\boldsymbol{S}_k) = 1$, for any $k = 1, \dots, N$. If one or more of the connected covers are non-minimal connected covers, the LS attempts to enhance the quality of the solution (and the fitness of the chromosome) by converting these connected covers to minimal connected covers. This step is carried out as follows. For each non-minimal connected cover $\boldsymbol{S}_k$, the LS procedure *prunes* $\boldsymbol{S}_k$ by removing redundant deployment points and hence converting $\boldsymbol{S}_k$ to a minimal connected cover denoted by $\boldsymbol{S}_k^m$. A redundant deployment point in $\boldsymbol{S}_k$ is a deployment point whose removal from the connected cover will not compromise its coverage or connectivity. For example, consider the chromosome illustrated in Fig. 4.3. The second connected cover represented in the chromosome, $\boldsymbol{S}_2 = \{d_2, d_4, d_6\}$ is a non-minimal connected cover because the deployment point d_6 is completely redundant. The LS identifies redundant deployment points by examining the tolerable failure combinations of SNs in the *off*-mode.

Table 4.1 Pseudo-code of the LS procedure in the proposed MA

Procedure LOCAL_SEARCH

Input: $c(N), \mathcal{F}(c(N))$. Decode input: $c(N) \leftrightarrow \boldsymbol{\mathcal{S}} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_N\},\ 1 \le N \le N_{UB}$, $R(\boldsymbol{\mathcal{S}})$
Initialize: $c_{LS}(N) \leftarrow c(N), \mathcal{F}\big(c_{LS}(N)\big) \leftarrow \mathcal{F}(c(N))$, $\boldsymbol{\mathcal{S}}_{LS} \leftarrow \boldsymbol{\mathcal{S}}$, $R(\boldsymbol{\mathcal{S}}_{LS}) \leftarrow R(\boldsymbol{\mathcal{S}})$
$\boldsymbol{\mathcal{S}}_{temp} \leftarrow \boldsymbol{\mathcal{S}}_{LS}$, $R(\boldsymbol{\mathcal{S}}_{temp}) \leftarrow R(\boldsymbol{\mathcal{S}}_{LS})$, $\boldsymbol{\mathcal{S}}'_{prune} \leftarrow \emptyset$
For $k = 1{:}N$
 If $\Phi(\boldsymbol{S}_k) \neq 0$, i.e. $\boldsymbol{S}_k$ is a non-minimal connected cover
  Prune $\boldsymbol{S}_k$ until there is no redundancy. Denote pruned $\boldsymbol{S}_k$ by $\boldsymbol{S}_k^m$
  $\boldsymbol{\mathcal{S}}'_{prune} \leftarrow \boldsymbol{\mathcal{S}}'_{prune} \cup [\boldsymbol{S}_k - \boldsymbol{S}_k^m]$
  Update $\boldsymbol{\mathcal{S}}_{temp}$: $\boldsymbol{S}_k \leftarrow \boldsymbol{S}_k^m$
 End **If**
End **For**
Update $R(\boldsymbol{\mathcal{S}}_{temp})$
If $R(\boldsymbol{\mathcal{S}}_{temp}) \ge R_{min}$
 Update $c_{LS}(N)$: $\boldsymbol{\mathcal{S}}_{LS} \leftarrow \boldsymbol{\mathcal{S}}_{temp}$, $R(\boldsymbol{\mathcal{S}}_{LS}) \leftarrow R(\boldsymbol{\mathcal{S}}_{temp})$. Set genes corresponding to $\boldsymbol{\mathcal{S}}'_{prune}$ to 0
Else (i.e. if reliability constraint is not met)
 Let incomplete SN sets in $c(N)$, if any, be denoted $\boldsymbol{\mathcal{S}}'_l$, where $1 \le l \le (N_{UB} - N)$
 For $l = 1{:}(N_{UB} - N)$
  Augment $\boldsymbol{\mathcal{S}}'_l$: $\boldsymbol{\mathcal{S}}'_l \leftarrow \boldsymbol{\mathcal{S}}'_l \cup \boldsymbol{\mathcal{S}}'_{prune}$. Test functionality of augmented SN set $\boldsymbol{\mathcal{S}}'_l$
  If $\boldsymbol{\mathcal{S}}'_l$ after augmentation became a connected cover, i.e. $\boldsymbol{\mathcal{S}}_{N+1} \leftarrow \boldsymbol{\mathcal{S}}'_l$
   Prune $\boldsymbol{\mathcal{S}}_{N+1}$ until there are no redundant deployment points. Denote pruned $\boldsymbol{\mathcal{S}}_{N+1}$ by $\boldsymbol{\mathcal{S}}_{N+1}^m$
   Update $\boldsymbol{\mathcal{S}}_{temp}$: $\boldsymbol{\mathcal{S}}_{N+1} \leftarrow \boldsymbol{\mathcal{S}}_{N+1}^m$, $\boldsymbol{\mathcal{S}}_{temp} \leftarrow \boldsymbol{\mathcal{S}}_{temp} \cup \boldsymbol{\mathcal{S}}_{N+1}$
   Break **For**
  End **If**
 End **For**
 If $\boldsymbol{\mathcal{S}}_{N+1} \neq \emptyset$ (i.e. if one of the incomplete SN sets became a connected cover)
  Update $R(\boldsymbol{\mathcal{S}}_{temp})$
  Update $c_{LS}(N)$: $\boldsymbol{\mathcal{S}}_{LS} \leftarrow \boldsymbol{\mathcal{S}}_{temp}$, $R(\boldsymbol{\mathcal{S}}_{LS}) \leftarrow R(\boldsymbol{\mathcal{S}}_{temp})$. Set genes corresponding to $[\boldsymbol{\mathcal{S}}_{N+1} - \boldsymbol{\mathcal{S}}_{N+1}^m]$ to 0
 Else
  Update $c_{LS}(N)$: $\boldsymbol{\mathcal{S}}_{LS} \leftarrow \boldsymbol{\mathcal{S}}_{temp}$, $R(\boldsymbol{\mathcal{S}}_{LS}) \leftarrow R(\boldsymbol{\mathcal{S}}_{temp})$. Set genes corresponding to $\boldsymbol{\mathcal{S}}'_{prune}$ to 0
 End **If**
End **If**
Update $\mathcal{F}\big(c_{LS}(N)\big)$
If $\mathcal{F}\big(c_{LS}(N)\big) \ge \mathcal{F}(c(N))$
 $c(N) \leftarrow c_{LS}(N),\ \mathcal{F}(c(N)) \leftarrow \mathcal{F}\big(c_{LS}(N)\big)$
End **If**
Output: $c(N), \mathcal{F}(c(N))$

produced by the search algorithm used to calculate $R(\boldsymbol{S}_k)$ described in Section 3.5.3.3 in Chapter 3. We denote the set of all the pruned deployment points by $\boldsymbol{\mathcal{S}}'_{prune}$.

After the application of the second step of the LS procedure, all the connected covers belonging to $\boldsymbol{\mathcal{S}}$ are guaranteed to be minimal connected covers. The updated solution is denoted by $\boldsymbol{\mathcal{S}}_{temp}$ and the updated combined reliability is denoted by $R(\boldsymbol{\mathcal{S}}_{temp})$. Note that if all the connected covers in $\boldsymbol{\mathcal{S}}$ are minimal, the second step of the LS procedure will affect no

change in $\boldsymbol{S}$ or $R(\boldsymbol{S})$, i.e. $\boldsymbol{S}_{temp} = \boldsymbol{S}$ and $R(\boldsymbol{S}_{temp}) = R(\boldsymbol{S})$. In the third step of the LS procedure (lines $11 - 30$), $R(\boldsymbol{S}_{temp})$ is compared to the specified minimum reliability R_{min}. If the reliability constraint is met, i.e. $R(\boldsymbol{S}_{temp}) \geq R_{min}$, no further steps are performed on the solution. Accordingly, the resulting chromosome of the LS $c_{LS}(N)$ is updated to reflect the pruning applied to the non-minimal connected covers in the second step. On the other hand, if the reliability constraint is not fulfilled, the LS attempts to further enhance the quality of the solution by attempting to construct an additional minimal connected cover, i.e. $\boldsymbol{S}_{N+1}$, to increase the combined reliability of the solution. This step is carried out as follows. For every SN set represented in $c(N)$ which does not amount to a connected cover, denoted by $\boldsymbol{S}'_l$ for $l = 1, \dots, (N_{UB} - N)$, the LS procedure checks if augmenting any of these SN sets by the pruned deployment points in the set $\boldsymbol{S}'_{prune}$ would result in an additional connected cover. If an additional connected cover was successfully constructed, it is pruned to a minimal connected cover $\boldsymbol{S}^m_{N+1}$ (if necessary) and added to the updated solution $\boldsymbol{S}_{temp}$. The resulting chromosome of the LS procedure $c_{LS}(N)$ is then updated to reflect the changes made to $\boldsymbol{S}_{temp}$. On the other hand, if none of the SN sets was successfully converted to a connected cover, $\boldsymbol{S}_{temp}$ remains unchanged and $c_{LS}(N)$ is updated accordingly. In the fourth and final step (lines $31 - 35$), the fitness of the resulting chromosome $\mathcal{F}(c_{LS}(N))$ is updated after the possible changes made to $c_{LS}(N)$ in steps 2 and 3. It is then compared with the fitness of the input chromosome $\mathcal{F}(c(N))$. If the LS procedure produces an enhancement in fitness (i.e. the condition $\mathcal{F}(c_{LS}(N)) \geq \mathcal{F}(c(N))$ holds), then $c_{LS}(N)$ and $\mathcal{F}(c_{LS}(N))$ are returned to replace the input chromosome and its corresponding fitness. Otherwise, the input chromosome and its corresponding fitness are returned unchanged.

4.3.1.6. Termination Conditions

The proposed MA is terminated if one of possible termination conditions occurs. The first termination condition is the algorithm going through a predetermined maximum number of generations denoted by n_{max}. The second condition is the algorithm going through a predetermined number maximum number of generations with no enhancement in the value of the best fitness discovered by the algorithm denoted by n_{conv}. The second termination condition signals that the algorithm has indeed converged to a solution and no further enhancement of fitness can be expected.

4.3.1.7. Measures to Reduce Computational Cost

The fitness function $\mathcal{F}(c(N))$ of the proposed MA, presented in Section 4.3.1.2 and expressed in (4.6), is in essence a complex and computationally expensive function to evaluate. This stems primarily from the third term of the fitness function, which requires the calculation of the reliability of the connected covers represented by a given chromosome. The proposed MA needs to evaluate the fitness function $\mathcal{F}(c(N))$ for every generated chromosome, both before and after mutation and during the proposed LS procedure. This poses a computational challenge. This is because the algorithm needs to evaluate the reliability of a large number of connected covers with varying levels of SN redundancy. To address this computational challenge, we apply the following two measures to reduce the computational cost associated with the fitness function:

- We terminate the search for the paths of $\boldsymbol{S}_k$ carried out by the search algorithm (outlined in Table 3.3 in Chapter 3) if a complete redundancy is discovered in step 3 of the algorithm, i.e. if the set $\boldsymbol{F}_o^1 \neq \{\phi\}$. If this occurs, then $\Phi(\boldsymbol{S}_k)$ is set to unity. The corresponding reliability $R(\boldsymbol{S}_k)$ used to evaluated the fitness function in (4.6) is the

lower bound of the exact reliability value calculated using the paths set of $\boldsymbol{S}_k$ discovered by the search algorithm before the search terminates. Although this measure may decrease the fitness of the chromosome to which the connected cover $\boldsymbol{S}_k$ belongs, this potential decrease is less significant for connected covers with a low redundancy level since the calculated lower bound will be a good estimation for the exact value. On the other hand, if a given connected cover is a minimal connected cover, the reliability is calculated exactly.

- A list of every connected cover the MA comes across and their calculated reliability is kept over all the generations/iterations of the algorithm. Every time the reliability of a given connected cover needs to be evaluated, the list is checked to see if this connected cover has been encountered before. If it has, the stored reliability value is used thus saving the reliability re-calculation time.

4.3.2. Proposed ACO Algorithm

In this section, we present our proposed ACO approach for solving the MCRC-SDP. First, we discuss how the MCRC-SDP is represented as a connected graph for ACO application, i.e. define the *construction graph* of the problem. Then, the ants' tour construction procedure is described, including the ants' neighborhood definitions and heuristic information. This is followed by the formulation of the cost function used for evaluating the quality of the solutions obtained by the ants. We then describe the pheromone management scheme followed by the LS procedure which we propose to be coupled with the ACO algorithm to enhance the quality of the obtained solutions. Finally, we summarize the steps of the proposed algorithm.

4.3.2.1. Construction Graph

In any ACO algorithm designed to solve a given optimization problem, ants build solutions incrementally by executing randomized walks or *tours* through a connected graph $G(\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is the set of the graph's vertices and $\boldsymbol{E}$ is the set of all the edges between the vertices in $\boldsymbol{V}$. Therefore, the first step in designing an ACO algorithm to solve a given optimization problem is to represent the problem as a connected graph $G(\boldsymbol{V}, \boldsymbol{E})$ by defining the sets $\boldsymbol{V}$ and $\boldsymbol{E}$ in terms of the problem's variables. For the MCRC-SDP at hand, the ACO construction graph is identical to the problem's graph defined by the set of deployment points $\boldsymbol{D}$ and the location of the sink node in the RoI denoted by d_0. Hence, $\boldsymbol{V}$ corresponds to the set of deployment points and the sink node location (i.e. $\boldsymbol{V} \equiv \{d_0, \boldsymbol{D}\} = \{d_0, d_1, d_2, \dots, d_{|\boldsymbol{D}|}\}$) and $\boldsymbol{E}$ corresponds to the set of undirected arcs/links connecting the deployment points and the sink node in $\boldsymbol{V}$ with each other.

4.3.2.2. Tour Construction

The ants' search behavior in a given construction graph is primarily influenced by a probabilistic transition rule, which controls how each ant selects its next vertex (i.e. deployment point) to visit during the construction of its tour (i.e. its solution to the problem). The probabilistic transition rule is in turn defined by three elements: the neighborhood definition(s), the heuristic information used by the ant and the pheromone trail values between the vertices of the construction graph. In this section we will discuss the first two elements while the pheromone management is discussed in Section 4.3.2.5.

A. Basic Idea:

Each ant a, $a = 1, \dots, m$, starts its tour at the sink node location d_0, which is an arbitrary location inside the boundaries of the RoI. Let the solution to the problem at hand which corresponds to the ant's tour be denoted by $\boldsymbol{S}^a$, initialized by an empty superset, i.e. $\boldsymbol{S}^a = \varphi$. Ant a then starts constructing a solution to the problem by consecutively building connected covers through transitioning among the deployment points in the construction graph. Let the index of the connected covers built by ant a be denoted by k, where $k = 1$ in the beginning of the ant's tour. An SN is deployed at each deployment point visited by ant a and the deployment point is added to the connected cover that ant a is currently building, denoted by $\boldsymbol{S}_k^a$. The connectivity of $\boldsymbol{S}_k^a$ to the sink node is maintained in each ant's transition by selectively defining the neighborhood of the ant's probabilistic transition rule (i.e. the candidate deployment points selected for the next transition), which will be discussed in the next sub-section. The building of $\boldsymbol{S}_k^a$ concludes when complete coverage of the target points in set $\boldsymbol{T}$ is achieved. The completed connected cover $\boldsymbol{S}_k^a$ is then added to the ant's solution superset $\boldsymbol{S}^a$. To check if the ant's tour is complete, $R(\boldsymbol{S}^a)$ is calculated using (4.4) and compared to the given minimum reliability level R_{min}. If $R(\boldsymbol{S}^a) \geq R_{min}$, then ant a's tour is concluded. Otherwise, the index k is incremented and ant a starts building a new connected cover through transitioning between the deployments points in the construction graph, excluding the points belonging to the connected cover(s) the ant built and added to $\boldsymbol{S}^a$ so far. Ant a continues building connected covers until $R(\boldsymbol{S}^a)$ meets or exceeds R_{min}. At this point the solution corresponding to ant a's tour is denoted $\boldsymbol{S}^a = \{\boldsymbol{S}_1^a, \boldsymbol{S}_2^a, \dots, \boldsymbol{S}_{N_a}^a\}$.

B. Heuristic Information and Neighborhood Definitions

At each tour construction step, ant a applies a probabilistic transition rule to select which deployment point it will visit next. The probability that ant a, currently at deployment point $d_i, i = 0,1, \dots, |\boldsymbol{D}|$, will select deployment point d_j, $j = 1,2, \dots |\boldsymbol{D}|$, to visit next is given by:

$$p_{ij}^a = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_j^a]^\beta}{\sum_{d_l \in \mathcal{N}_i^a} [\tau_{il}]^\alpha [\eta_l^a]^\beta} \ , & if\ d_j \in \boldsymbol{\mathcal{N}}_i^a \\ 0 \ , & otherwise \end{cases}, \tag{4.7}$$

where τ_{ij} is the pheromone trail value between deployment points d_i (or sink node if $i = 0$ at the beginning of the tour) and d_j, η_j^a is the heuristic value of adding the deployment point d_j to the connected cover currently being built by ant a, i.e. $\boldsymbol{S}_k^a$, $\boldsymbol{\mathcal{N}}_i^a$ is the feasible neighborhood of ant a at its current position in the construction graph at d_i, and α and β are the parameters that control the influence of the pheromone trail values and heuristic information on p_{ij}^a, respectively.

The definition of the feasible neighborhood $\boldsymbol{\mathcal{N}}_i^a$ of ant a at a given current position d_i depends on whether the next transition the ant is making is an *intra-connected cover transition* or *inter-connected cover transition*. Ant a makes an intra-connected cover transition when the current connected cover its building, i.e. $\boldsymbol{S}_k^a$, is not yet complete after the addition of the deployment point d_i at which the ant is currently present, i.e. $\boldsymbol{S}_k^a \neq \varphi$. On the other hand, ant a makes an inter-connected cover transition when its previous transition has completed $\boldsymbol{S}_k^a$ but its tour is not yet complete, i.e. $R(\boldsymbol{S}^a) < R_{min}$. In this case, the next transition of a is the start of a new connected cover, i.e. $k = k + 1$ and $\boldsymbol{S}_k^a = \varphi$.

For an intra-connected cover transition, the feasible neighborhood $\boldsymbol{\mathcal{N}}_i^a$ of ant a at a given current position d_i is defined as follows:

$$\boldsymbol{\mathcal{N}}_i^a = \begin{cases} \boldsymbol{\mathcal{N}}_{ieff}^a, \boldsymbol{\mathcal{N}}_{ieff}^a \neq \varphi \\ \boldsymbol{\mathcal{N}}_{ifull}^a, \boldsymbol{\mathcal{N}}_{ieff}^a = \varphi \end{cases}, \tag{4.8}$$

where $\boldsymbol{\mathcal{N}}_{ifull}^a$ is defined as the set of deployment points within the communication range r_c of any deployment point belonging to $\boldsymbol{S}_k^a$. Let the set $\boldsymbol{D}^-$ be the set of deployment points not visited so far by ant a in its current tour. The set $\boldsymbol{\mathcal{N}}_{ifull}^a$ can then be expressed as follows:

$$\boldsymbol{\mathcal{N}}_{ifull}^a = \{d_j \in \boldsymbol{D}^- : \left\|d_j d_{j'}\right\| \leq r_c, \text{ for any } d_{j'} \in \boldsymbol{S}_k^a\}, \tag{4.9}$$

The set $\boldsymbol{\mathcal{N}}_{ieff}^a$, on the other hand, is a subset of deployment points belonging to $\boldsymbol{\mathcal{N}}_{ifull}^a$ that would offer a *coverage gain* for $\boldsymbol{S}_k^a$, i.e. the addition of any of the deployment points belonging to $\boldsymbol{\mathcal{N}}_{ieff}^a$ to $\boldsymbol{S}_k^a$ would result in the coverage of uncovered target points in $\boldsymbol{T}$ by $\boldsymbol{S}_k^a$. Let the coverage gain of a deployment point $d_j \in \boldsymbol{\mathcal{N}}_{ifull}^a$ be denoted by $\mathcal{g}_j^a$. We define the coverage gain $\mathcal{g}_j^a$ as the number of uncovered target points by $\boldsymbol{S}_k^a$ that would be covered if an SN is deployed at d_j, i.e. if d_j is added to the current connected cover $\boldsymbol{S}_k^a$. Hence, the set $\boldsymbol{\mathcal{N}}_{ieff}^a$ can be expressed as follows:

$$\boldsymbol{\mathcal{N}}_{ieff}^a = \{d_j \in \boldsymbol{\mathcal{N}}_{ifull}^a : \mathcal{g}_j^a \neq 0\}, \tag{4.10}$$

For an inter-connected cover transition, on the other hand, the feasible neighborhood $\boldsymbol{\mathcal{N}}_i^a$ of ant a at a given current position d_i is defined as:

$$\boldsymbol{\mathcal{N}}_i^a = \boldsymbol{\mathcal{N}}_{sink}^a, \tag{4.11}$$

where $\boldsymbol{\mathcal{N}}_{sink}^a$ is defined as the set of deployment points belonging to $\boldsymbol{D}^-$ which are within a distance equal to the SN communication range r_c. Note that at the beginning of the tour, $i = 0$ and $\boldsymbol{D}^- = \boldsymbol{D}$. Accordingly, we can express $\boldsymbol{\mathcal{N}}_{sink}^a$ as follows:

$$\boldsymbol{\mathcal{N}}_{sink}^a = \{d_j \in \boldsymbol{D}^- : \left\|d_j d_0\right\| \leq r_c\}, \tag{4.12}$$

The neighborhood definitions in (4.8) and (4.11) are designed to achieve two goals. The first goal is to guarantee the connectivity of each cover built by ant a. Since all ants start their tours at d_0, the neighborhood definitions guarantee that each added deployment point to $\boldsymbol{S}_k^a$ will be connected to the sink node via single or multi-hop communication. The second goal is to minimize the probability of adding redundant deployment points to any of the connected covers built by the ants, i.e. minimize the probability of ants constructing tours that correspond to infeasible solutions to the MCRC-SDP that violate the redundancy constraint expressed in (4.5). This goal is achieved specifically through the neighborhood definition in (9). The neighborhood definition restricts the candidate deployment points for the ant's next transition to points which belong to the set $\boldsymbol{\mathcal{N}}_{ifull}^a$ and have a non-zero coverage gain, i.e. $\boldsymbol{\mathcal{N}}_{ieff}^a$. In the case where $\boldsymbol{\mathcal{N}}_{ieff}^a = \varphi$, however, adding a redundant deployment point to $\boldsymbol{S}_k^a$ may occur.

The heuristic value of adding deployment point d_j to the current connected cover $\boldsymbol{S}_k^a$ being built by ant a, denoted by η_j^a, is directly proportional to its coverage gain $\mathcal{g}_j^a$ and is defined as:

$$\eta_j^a = \mathcal{g}_j^a + 1 \tag{4.13}$$

Equation (4.13) applies to both types of ant's transitions, namely, the intra- and inter-connected cover transitions, where in the latter case the uncovered target points are the entire set $\boldsymbol{T}$, since the current connected cover $\boldsymbol{S}_k^a$ in this case is empty, i.e. $\boldsymbol{S}_k^a = \varphi$. Table 4.2 summarizes the ants' tour construction procedure.

Table 4.2 Pseudo code of the tour construction procedure in the proposed ACO algorithm

Procedure TOUR_CONSTRUCTION	
1	Input: $\boldsymbol{D}$, $\boldsymbol{T}$, d_0, R_{min}, , λ, τ_{ij} for $i = 0,1,\dots,\lvert\boldsymbol{D}\rvert$, $j = 1,2,\dots,\lvert\boldsymbol{D}\rvert$
2	Initialize: $\boldsymbol{\mathcal{S}}^a = \varphi$, $R(\boldsymbol{\mathcal{S}}^a) = 0$, $k = 0$, $\boldsymbol{D}^- = \boldsymbol{D}$, ant starts tour at $d_0 (i = 0)$
3	**While** $R(\boldsymbol{\mathcal{S}}^a) < R_{min}$
4	Build a new connected cover: $k \leftarrow k + 1$, $\boldsymbol{S}_k^a = \varphi$, $\boldsymbol{T}_{cov} = \varphi$
5	**While** $\boldsymbol{T}_{cov} \neq \boldsymbol{T}$ (i.e. $\boldsymbol{S}_k^a$ is not a complete connected cover)
6	Identify $\boldsymbol{\mathcal{N}}_i^a$ using (9) and (12)
7	Calculate coverage gain $\mathcal{g}_j \forall d_j \in \boldsymbol{\mathcal{N}}_i^a$
8	Apply transition rule in (8) to choose next deployment point
9	Update $\boldsymbol{S}_k^a$
10	Update $\boldsymbol{T}_{cov}$ (i.e. update coverage of $\boldsymbol{S}_k^a$)
11	**End While**
12	Update $\boldsymbol{\mathcal{S}}^a$: $\boldsymbol{\mathcal{S}}^a \leftarrow \{\boldsymbol{\mathcal{S}}^a, \boldsymbol{S}_k^a\}$
13	Calculate $R(\boldsymbol{S}_k^a)$ and Update $R(\boldsymbol{\mathcal{S}}^a)$
14	Update $\boldsymbol{D}^-$: $\boldsymbol{D}^- \leftarrow \boldsymbol{D}^- - \boldsymbol{S}_k^a$
15	**End While**
16	Output: $\boldsymbol{\mathcal{S}}^a = \{\boldsymbol{S}_1^a, \boldsymbol{S}_2^a, \dots, \boldsymbol{S}_{N_a}^a\}$, $R(\boldsymbol{\mathcal{S}}^a)$

4.3.2.3. Cost Function

To evaluate the quality of the solution to the MCRC-SDP corresponding to the tour constructed by ant a , i.e. $\boldsymbol{\mathcal{S}}^a = \{\boldsymbol{S}_1^a, \boldsymbol{S}_2^a, \dots, \boldsymbol{S}_{N_a}^a\}$, the following cost function is used:

$$\mathcal{C}(\boldsymbol{\mathcal{S}}^a) = \omega_1 \sum_{k=1}^{N_a} |\boldsymbol{S}_k| + \omega_2 \sum_{k=1}^{N_a} \Phi(\boldsymbol{S}_k), \tag{4.14}$$

where the first term of the cost function, $\sum_{k=1}^{N_a} |\boldsymbol{S}_k| = |\boldsymbol{\mathcal{S}}^a|$, represents the total number of deployment points (i.e. deployed SNs) belonging to the N_a connected covers in $\boldsymbol{\mathcal{S}}^a$ multiplied by a constant weight ω_1. The second term of the cost function penalizes *every* connected cover that contains complete redundancy i.e. that is not a minimal connected cover by a penalty equal to the constant weight ω_2.

Since the objective of the MCRC-SDP is to minimize the total deployment cost of the network, i.e. minimize $\sum_{k=1}^{N_a} |\boldsymbol{S}_k| = |\boldsymbol{\mathcal{S}}^a|$, the weights ω_1 and ω_2 are set such that the cost assigned to the solutions follow the following criterion. All the solutions which meet both the reliability and the redundancy constraints expressed in (4.4) and (4.5), respectively, have a lower cost than all the solutions that meet the reliability constraint but fail to meet the redundancy constraint, i.e. solutions that have one or more non-minimal connected covers. As such, if ω_1 is set to unity such that the first term of the cost function is equal to the total number of deployed SNs (i.e. the deployment cost), then ω_2 must be greater than $|\boldsymbol{D}|$ (since the maximum value of $|\boldsymbol{\mathcal{S}}^a|$ is $|\boldsymbol{D}|$). Accordingly, we set $\omega_1 = 1$ and $\omega_2 = |\boldsymbol{D}| + 1$.

4.3.2.4. Local Search Procedure

As stated earlier, the proposed ACO algorithm for solving the MCRC-SDP is coupled with an LS procedure that helps the algorithm find higher quality solutions to the problem. Similar to the study in [119], after the ants have completed the construction of their tours/solutions in every ieteration, the LS procedure is applied to each of the constructed solutions with the objective of reducing its cost as evaluated by the cost function in (4.14). Table 4.3 shows the pseudo code of the proposed LS procedure.

The operation of the LS procedure can be described as follows. Assuming the LS is applied on the solution $\boldsymbol{\mathcal{S}}^a = \{\boldsymbol{S}_1^a, \boldsymbol{S}_2^a, \dots, \boldsymbol{S}_{N_a}^a\}$ constructed by ant a, the first step of the LS procedure is to determine whether any of the connected covers in $\boldsymbol{\mathcal{S}}^a$ violates the redundancy constraint in (4.5), i.e. $\Phi(\boldsymbol{S}_k^a) = 1$, for any $k = 1, \dots, N_a$. If all the connected covers are minimal connected covers, i.e. $\boldsymbol{\mathcal{S}}^a$ is a feasible solution, the LS procedure returns $\boldsymbol{\mathcal{S}}^a$ and its corresponding reliability $R(\boldsymbol{\mathcal{S}}^a)$ unchanged. On the other hand, if one or more of the connected covers in $\boldsymbol{\mathcal{S}}^a$ have redundant deployment points, the LS attempts to reduce the cost $\mathcal{C}(\boldsymbol{\mathcal{S}}^a)$ by converting these connected covers to minimal connected covers. This procedure is carried out as follows. For each non-minimal connected cover $\boldsymbol{S}_k^a$, the LS procedure *prunes* $\boldsymbol{S}_k^a$ by removing completely redundant deployment points in the same method used in the LS procedure in the proposed MA presented in Section 4.3.1.5. Let the pruned connected cover be denoted $\boldsymbol{S}_{kp}^a$. The LS procedure then updates the combined reliability of $\boldsymbol{\mathcal{S}}^a$ accordingly (i.e. substituting $R(\boldsymbol{S}_k^a)$ with $R(\boldsymbol{S}_{kp}^a)$ in (4.4)). If the updated combined reliability of $\boldsymbol{\mathcal{S}}^a$ exceeds or meets R_{min}, the pruned connected cover $\boldsymbol{S}_{kp}^a$ replaces $\boldsymbol{S}_k^a$ in the solution $\boldsymbol{\mathcal{S}}^a$, otherwise $\boldsymbol{S}_k^a$ is kept without change in $\boldsymbol{\mathcal{S}}^a$. The same above steps are repeated for every non-minimal connected cover in $\boldsymbol{\mathcal{S}}^a$. Accordingly, for every pruned connected cover that replaces a non-minimal connected cover in $\boldsymbol{\mathcal{S}}^a$, the cost $\mathcal{C}(\boldsymbol{\mathcal{S}}^a)$ is reduced by the value of $\omega_2 = |\boldsymbol{D}| + 1$.

Table 4.3 Pseudo code of the LS procedure for the proposed ACO algorithm

Procedure LOCAL_SEARCH

Input: $\boldsymbol{\mathcal{S}}^a = \{\boldsymbol{S}_1^a, \boldsymbol{S}_2^a, \dots, \boldsymbol{S}_{N_a}^a\}$, $\mathcal{C}(\boldsymbol{\mathcal{S}}^a)$, $R(\boldsymbol{\mathcal{S}}^a)$
Initialize: $\mathcal{C}_{LS}(\boldsymbol{\mathcal{S}}^a) \leftarrow \mathcal{C}(\boldsymbol{\mathcal{S}}^a)$, $\boldsymbol{\mathcal{S}}^a{}_{LS} \leftarrow \boldsymbol{\mathcal{S}}^a$, $R(\boldsymbol{\mathcal{S}}^a{}_{LS}) \leftarrow R(\boldsymbol{\mathcal{S}}^a)$
$\boldsymbol{\mathcal{S}}^a{}_{temp} \leftarrow \boldsymbol{\mathcal{S}}^a{}_{LS}$, $R(\boldsymbol{\mathcal{S}}^a{}_{temp}) \leftarrow R(\boldsymbol{\mathcal{S}}^a{}_{LS})$
For $k = 1, \dots, N_a$
  If $\Phi(\boldsymbol{S}_k^a) = 1$, i.e. if $\boldsymbol{S}_k^a$ is *not* a minimal connected cover
    Prune $\boldsymbol{S}_k^a$ until there are no redundant deployment points. Let pruned $\boldsymbol{S}_k^a$ be denoted $\boldsymbol{S}_{kp}^a$
    Update $\boldsymbol{\mathcal{S}}^a{}_{temp}$: $\boldsymbol{S}_k^a \leftarrow \boldsymbol{S}_{kp}^a$
    Update $R(\boldsymbol{\mathcal{S}}^a{}_{temp})$
    If $R(\boldsymbol{\mathcal{S}}^a{}_{temp}) \geq R_{min}$
      Update $\boldsymbol{\mathcal{S}}^a{}_{LS}$: $\boldsymbol{S}_k^a \leftarrow \boldsymbol{S}_{kp}^a$. $R(\boldsymbol{\mathcal{S}}^a{}_{LS}) \leftarrow R(\boldsymbol{\mathcal{S}}^a{}_{temp})$
      $\mathcal{C}_{LS}(\boldsymbol{\mathcal{S}}^a) \leftarrow \mathcal{C}_{LS}(\boldsymbol{\mathcal{S}}^a) - \omega_2$
    Else
      $\boldsymbol{\mathcal{S}}^a{}_{LS}$ remains unchanged → $\mathcal{C}_{LS}(\boldsymbol{\mathcal{S}}^a)$ remains unchanged
      $\boldsymbol{\mathcal{S}}^a{}_{temp} \leftarrow \boldsymbol{\mathcal{S}}^a{}_{LS}$, $R(\boldsymbol{\mathcal{S}}^a{}_{temp}) \leftarrow R(\boldsymbol{\mathcal{S}}^a{}_{LS})$
    End If
  End If
End For
Output: $\mathcal{C}_{LS}(\boldsymbol{\mathcal{S}}^a)$, $\mathcal{C}_{LS}(\boldsymbol{\mathcal{S}}^a)$

4.3.2.5. Pheromone Management

After all the ants have constructed their tours and the LS procedure has been applied to the corresponding solutions, pheromone trail values are updated according to the MAX-MIN Ant System (MMAS) [68] updating rule which can be expressed as follows:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \Delta\tau_{ij}^{ib}, \tag{4.15}$$

where $i = 0, 1, \ldots, |\boldsymbol{D}|$, $j = 1, \ldots, |\boldsymbol{D}|$, $\rho \in (0,1)$ is the pheromone evaporation factor and the added pheromone trail $\Delta\tau_{ij}^{ib}$ can be given by the following equation:

$$\Delta\tau_{ij}^{ib} = \begin{cases} 1/\mathcal{C}^{ib}, & if\, d_j \in \boldsymbol{S}^{ib} \\ 0, & otherwise \end{cases}, \tag{4.16}$$

where $\boldsymbol{S}^{ib}$ is the best solution found by the ants in the current iteration of the algorithm (i.e. iteration-best solution) and $\mathcal{C}^{ib}$ is its cost evaluated by the cost function expressed in (4.13). According to the MMAS pheromone update rule, only the ant which found the solution with the highest quality (i.e. the lowest cost) gets to deposit pheromone on the arcs of the construction graph.

Note that pheromone is deposited on all the arcs leading to the deployment point $d_j \in \boldsymbol{S}^{ib}$. This is because the proposed algorithm rewards the inclusion of a deployment point in the iteration-best solution, regardless of its position in the solution (i.e. regardless of the connected cover to which it belongs). The reasoning behind this is that the inclusion of such advantageous deployment points in different connected covers can lead to different but equally good solutions to the problem. Thus, exploring different permutations of these deployment points is essential to finding high quality solutions.

Since the MMAS pheromone update rule strongly exploits the best solution found in each iteration, upper and lower limits, denoted τ_{max} and τ_{min}, are imposed on the pheromone trail value on each arc of the construction graph. This strategy is called *pheromone constraining* and is followed to avoid a stagnation situation where the algorithm converges prematurely to good but sub-optimal solutions. This is due to the excessive increase of the pheromone trails on the arcs leading to the deployment points belonging to those solutions. Pheromone constraining ensures that the probability of an ant a on deployment point d_i selecting a deployment point $d_j \in \boldsymbol{\mathcal{N}}_i^a$ is always greater than zero. The value of τ_{max} is given by:

$$\tau_{max} = 1/\rho\mathcal{C}^{bs}, \tag{4.17}$$

where $\mathcal{C}^{bs}$ is the best solution found so far by the algorithm (i.e. best-so-far solution). Note that every time a higher quality solution is found and $\mathcal{C}^{bs}$ is updated, the value of τ_{max} is updated accordingly. On the other hand, the value of τ_{min} is given by (4.18), where $\mathscr{b}$ is a constant that is set by experimentation.

$$\tau_{min} = \tau_{max}/\mathscr{b}, \tag{4.18}$$

4.3.2.6. Summary of the proposed ACO algorithm

Table 4.4 summarizes the different steps in the proposed ACO algorithm for solving the MCRC-SDP. The input to the proposed ACO algorithm includes all the MCRC-SDP instance parameters ($\boldsymbol{D}, \boldsymbol{T}, d_0, \lambda, R_{min}, N_{UB}, r_s, r_c$) and the ACO related parameters (m, ρ, it_{max}, it_c). The ACO parameters it_{max} and it_c are defined as the maximum allowed number of iterations the algorithm can carry out and the number of successive iterations the algorithm can carry out with no enhancement in the best so far solution cost $\mathcal{C}^{bs}$ before it is terminated, i.e. before it is decided that the algorithm has converged.

In the first step of the proposed algorithm, the best-so-far solution cost $\mathcal{C}^{bs}$ is initialized to a high value in order to ensure that it is replaced by the best solution cost found in the first iteration. All Pheromone trails are initialized to unity to ensure that they are constrained to the upper limit calculated at the end of the first iteration using (4.17). Then, each ant a, for $a = 1, \dots, m$, constructs its tour/ solution $\boldsymbol{\mathcal{S}}^a$ according to the tour construction procedure presented in Section 4.3.2.2 and summarized in Table 4.2. The cost of ant a's solution $\mathcal{C}(\boldsymbol{\mathcal{S}}^a)$ is evaluated using (4.14). Then the LS procedure presented in Section 4.3.2.4 and summarized in Table 4.3 is applied to $\boldsymbol{\mathcal{S}}^a$. Note that if the LS procedure produced no reduction in the value of $\mathcal{C}(\boldsymbol{\mathcal{S}}^a)$, it returns the original solution and cost unaltered.

After these steps are applied for each ant, the iteration-best solution $\boldsymbol{\mathcal{S}}^{ib}$ and the corresponding cost $\mathcal{C}^{ib}$ are identified and used to update the pheromone trail values using (4.15) and (4.16). Next, the best-so-far solution is updated if $\mathcal{C}^{ib}$ is less than the current $\mathcal{C}^{bs}$ and the values of τ_{max} and τ_{min} are updated accordingly using (4.17) and (4.18). The pheromone constraining procedure follows as described in Section 4.3.2.5. Finally, the algorithm is terminated if it goes through n_{max} iterations or if it goes through n_{conv} iterations with no enhancement in the best-so-far solution cost $\mathcal{C}^{bs}$.

Table 4.4 Pseudo code of the proposed ACO algorithm

	Ant Colony Optimization Algorithm for Solving MCRC-SDP
1	Input: $\boldsymbol{D}, \boldsymbol{T}, d_0, R_{min}, \lambda, N_{UB}, r_s, r_c, m, \rho, n_{max}, n_{conv}$
2	Initialize: $it = 0$, $\mathcal{C}^{bs} = \infty, \boldsymbol{\mathcal{S}}^{bs} = \varphi, \tau_0 = 1$
3	**While** ($it < n_{max}$ & $n_{conv} > 0$)
4	Increment iterations counter: $it \leftarrow it + 1$
5	**For** $a = 1, \dots, m$
6	Apply TOUR_CONSTRUCTION in Table 4.2 procedure to build $\boldsymbol{\mathcal{S}}^a$
7	Calculate tour cost $\mathcal{C}(\boldsymbol{\mathcal{S}}^a)$ using (4.14)
8	Apply LOCAL_SEARCH procedure in Table 4.3: $\boldsymbol{\mathcal{S}}^a \leftarrow \boldsymbol{\mathcal{S}}^a{}_{LS}$, $\mathcal{C}(\boldsymbol{\mathcal{S}}^a) \leftarrow \mathcal{C}_{LS}(\boldsymbol{\mathcal{S}}^a)$, $R(\boldsymbol{\mathcal{S}}^a) \leftarrow R(\boldsymbol{\mathcal{S}}^a{}_{LS})$
9	**End For**
10	Identify iteration-best solution $\boldsymbol{\mathcal{S}}^{ib}$ and cost $\mathcal{C}^{ib}$
11	Update pheromone trails using (4.15) , (4.16)
12	**If** $\mathcal{C}^{ib} < \mathcal{C}^{bs}$
13	Update best solution so far: $\boldsymbol{\mathcal{S}}^{bs} \leftarrow \boldsymbol{\mathcal{S}}^{ib}$, $\mathcal{C}^{bs} \leftarrow \mathcal{C}^{ib}$
14	Re-initialize convergence counter n_{conv} to starting value
15	**Else**
16	Decrement convergence counter: $n_{conv} \leftarrow n_{conv} - 1$
17	**End If**
18	Apply Pheromone constraining to τ_{max} and τ_{min} (4.17) , (4.18)
19	**End While**
20	Output: $\boldsymbol{\mathcal{S}}^{bs} = \{\boldsymbol{\mathcal{S}}_1^{bs}, \boldsymbol{\mathcal{S}}_2^{bs}, \dots, \boldsymbol{\mathcal{S}}_{N_{bs}}^{bs}\}$, $\mathcal{C}^{bs}$

4.3.2.7. Measures to Reduce Computational Cost

The same two measures discussed in Section 4.3.1.7, which are applied in the proposed MA, are applied in the proposed ACO algorithm to reduce the computational cost associated with the cost function expressed in (4.14).

4.4. Experimental Results and Discussion

In this section, the performance of the proposed MA and ACO algorithms in solving the MCRC-SDP expressed in (4.1) - (4.5) is evaluated in terms of two main performance metrics, namely, the quality of the obtained solutions and the computational cost. Since, to the best of our knowledge, the proposed algorithms are the first algorithms to solve the MCRC-SDP, we benchmark their performance using a simple GH which attempts to find feasible solutions of good quality to the MCRC-SDP by successively building connected covers until the reliability constraint is met.

4.4.1. Experimental Setup

For the conducted experiments, several instances of the MCRC-SDP of different scales and different values of the minimum reliability R_{min} are generated. We assume that the RoI is a two-dimensional square area equal to 100×100 m^2. The set of target points $\boldsymbol{T}$, the set of deployment points $\boldsymbol{D}$ and the location of the sink node d_0 are all generated randomly inside the perimeter of the RoI. The scale of the problem is identified by the sizes of the sets $\boldsymbol{D}$ and $\boldsymbol{T}$, denoted by $|\boldsymbol{D}|$ and $|\boldsymbol{T}|$ respectively. For each problem scale, the upper bound of the number of connected covers N_{UB} is calculated using the procedure presented in Section 4.2.3. We denote each problem scale a *test case*. Since the value of R_{min} affects the difficulty level of the problem instance (the higher the value the more difficult it is to solve the problem instance), three values for R_{min} are considered, specifically $R_{min} = 0.99$, 0.999 and 0.9999, for each test case. That is, each test case generates three problem instances, one for each of the three R_{min} values. Table 4.5 shows the data pertinent to each test case, namely the values of $|\boldsymbol{D}|$, $|\boldsymbol{T}|$ and N_{UB}. For all problem instances, we assume the following SN-related parameters: disk-coverage model, sensing range $r_s = 30$ m, communication range $r_c = 50$ m and probabilities of failure of sensor, transceiver, processor and battery, $\lambda_s = 1.0\times 10^{-2}$, $\lambda_t = 5.0\times 10^{-3}$, $\lambda_p = 2.0 \times 10^{-3}$ and $\lambda_b = 1.0 \times 10^{-3}$, respectively.

Table 4.5 Data of test cases used to evaluate the proposed MA and ACO algorithm

Test Case	$\|\boldsymbol{D}\|$	$\|\boldsymbol{T}\|$	N_{UB}
TC1	30	15	4
TC2	40	25	4
TC3	50	35	5
TC4	60	45	5
TC5	70	55	6
TC6	80	65	6
TC7	90	75	7

4.4.2. Parameter Settings of the Proposed Algorithms

In this section, we discuss how the different parameters of the proposed MA and ACO algorithm are set.

4.4.2.1. Parameter Settings of the Proposed MA

Selecting the MA parameters, including the population size, the crossover rate, the mutation rate and the LS procedure scheduling, carefully is an important task in optimizing its performance in solving the optimization problem at hand [133]. However, it is also a difficult task since a wide range of numbers for each of these parameters have been recommended in the literature [134], i.e. there are no *optimal* parameter settings that can be applied to all optimization problems. Therefore, it is advised that general guidelines for GA parameters setting are followed along with an initial parameter sensitivity testing on the problem at hand, which reveals the parameter(s) by which the algorithm performance is most affected.

For the MCRC-SDP at hand, the initial parameter sensitivity testing revealed that the performance of proposed MA is most sensitive to the LS procedure schedule, especially in terms of the computational cost, which is measured by the total CPU run time in seconds for the algorithm to converge or perform the set maximum number of iterations. Since the ratio P_{LS} follows a gradual linear increase as shown in Fig. 4.4, the parameter n_{LS} controls the *speed* by which this increase occurs during the progress of the MA. According to the study in [132], this speed is highly dependent on the combination of both the optimization problem nature (i.e. objective function, constraints, and the design of the MA fitness function) and the used LS procedure. On one hand, decreasing the value of n_{LS} decreases the speed of the LS procedure application and hence decreases the overall number of chromosomes undergoing the procedure throughout a given number of iterations. This in turn results in a decrease in the additional computational cost of the LS procedure and may result in decreasing the overall computational cost/run time of the algorithm. On the other hand, decreasing the speed of the LS procedure application may also decrease the speed of convergence of the algorithm, i.e. increase the number of iterations the algorithm requires to converge and hence increase the overall computational cost/run time. Similarly, decreasing the value of $\boldsymbol{n_{LS}}$ increases the speed of the LS procedure application and its effect on the overall computational cost/ run time of the algorithm may be positive or negative as well. It may also cause the algorithm to quickly get trapped in local minima and converge prematurely. Hence, setting the value of the parameter n_{LS} requires tuning using experimental results. The remainder of the proposed MA parameters are set as follows [135], [136]: $\mu = 2|\boldsymbol{D}|$ (where $|\boldsymbol{D}|$ represents the length of the integer-encode chromosome), crossover rate is set to 1.0 and the mutation rate is set to $2/|\boldsymbol{D}|$.

In order to tune the LS procedure scheduling to produce the best performance of the proposed MA in terms of the quality of solutions and the computational cost, we selected two MCRC-SDP instances at random from the twenty one problem instances generated from the seven test cases in Table 4.5, namely, TC4 at $R_{min} = 0.99$ and TC5 at $R_{min} = 0.999$. For both selected problem instances, the proposed MA is applied using seven different values for n_{LS} , specifically, 5,10,20,30,40,50,100. For each value of n_{LS} ten independent runs of the proposed MA are performed to account for the algorithm's stochastic nature. Since this experiment involves a significantly large number of runs, we adopted the conventional two-mode SN model in all reliability calculations in this experiment. This significantly decreases the computational cost of calculating the fitness function expressed in (4.6) as can be inferred from the discussion in Section 3.6.2 in Chapter 3. This implicit simplification of the fitness function has no effect on the outcome of this experiment since the effect of the value of n_{LS} on the performance of the proposed MA is independent of the accuracy of the SN model adopted in reliability calculations. Results of the experiment are shown in Fig. 4.5 and Fig. 4.6.

Fig. 4.5 shows the quality of the obtained solutions, measured by the average number of deployment points in the solutions (i.e. the deployment cost) for TC4 at $R_{min} = 0.99$ and for TC5 at $R_{min} = 0.999$. Fig. 4.6 shows the computational cost, measured by the average CPU run time in seconds, for both problem instances. From Fig. 4.5 we see that the effect of varying the speed of the LS procedure application for the tested range of values for n_{LS} on the quality of the obtained solutions does not appear to be significant in both tested problem instances. However, Fig. 4.6 shows that in both tested problem instances, the lowest overall computational cost of applying the proposed MA occurs at the slowest LS procedure schedule, i.e. at $n_{LS} = 100$. At that setting, the rate of increase of the number of chromosomes undergoing the LS procedure per iteration, which is equal to $\lfloor \mu / n_{LS} \rfloor$, is equal

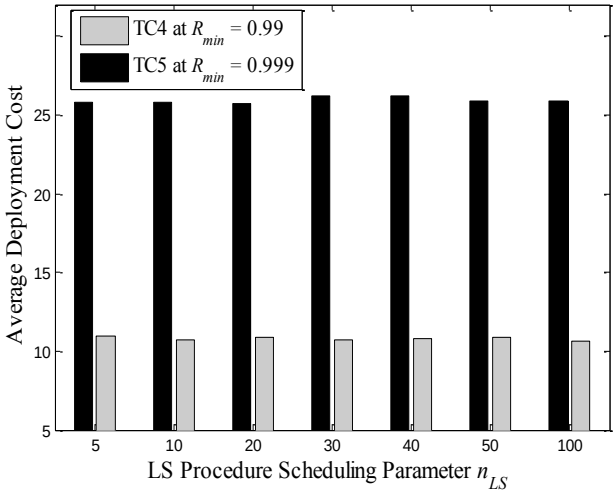


Fig. 4.5 Average deployment cost of the solutions obtained by the proposed MA for different LS procedure schedules (different values of n_{LS}) for TC4 at $R_{min} = 0.99$ and TC5 at $R_{min} = 0.999$.

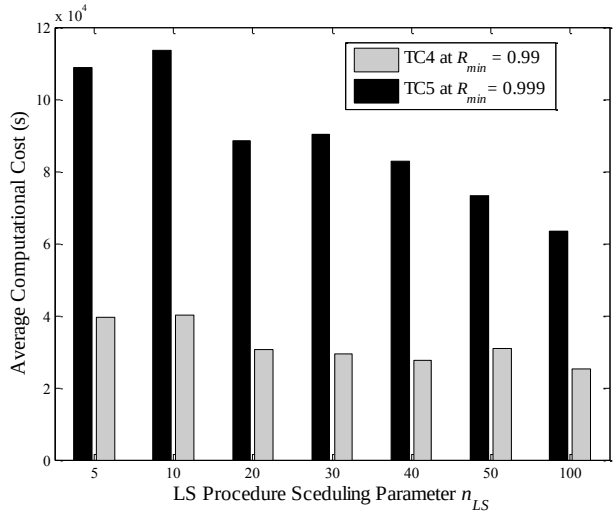


Fig. 4.6 Average computational cost , measured by the CPU run time in seconds, of the proposed MA for different LS procedure schedules (different values of n_{LS}) for TC4 at $R_{min} = 0.99$ and TC5 at $R_{min} = 0.999$.

to unity for both tested problem instances. This slow speed of LS procedure application progression does not negatively affect the ability of the solution to obtain relatively good quality solutions but significantly decrease the overall computational cost of the proposed

MA by decreasing the additional computational cost of the LS procedure. Hence, we will use $n_{LS} = 100$ for all tested problem instances in the MA experiments presented later in this chapter. Table 4.6 shows the settings of the different parameters of the proposed MA.

Table 4.6 Parameters of the proposed MA

Parameter	Setting
Population Size μ	$2\|\boldsymbol{D}\|$
Parent Selection	Roulette Wheel
Crossover Operator	Single-point with rate 1.0
Mutation Operator	Creep with rate $2/\|\boldsymbol{D}\|$
Survivor Selection	$\mu + \lambda$ scheme
Heuristics	Scattering + LS procedure in Table 4.1
LS Procedure Schedule n_{LS}	100
Maximum No. of Generations n_{max}	100
No. of Generations for Convergence n_{conv}	20

4.4.2.2. Parameters Setting of the Proposed ACO Algorithm

To set the different parameters of the proposed ACO algorithm, we follow the general guidelines in [68], which suggests that for a wide variety of optimization problems, the number of ants m and the pheromone evaporation rate ρ can be set to 30 and 0.5, to achieve optimal results. For a fair comparison with the proposed MA in the following experiments, we set $n_{max} = 100$ and $n_{conv} = 20$. However, the optimal values of the ACO parameters α and β vary depending on the problem at hand. It is therefore important to find their optimum configuration that would result in the best average solution quality obtained by the ACO algorithm for the MCRC-SDP. In the ACO literature, values of both α and β can vary between 1 and 5, with the optimum configuration largely depending on the type of problem the ACO algorithm is designed to solve and whether or not the algorithm is coupled with an LS procedure [68].

Similar to the method followed in tuning n_{LS} in the proposed MA, we selected two problem instances at random from the twenty one problem instances generated from the seven test cases in Table 4.5, namely test case TC3 for $R_{min} = 0.9999$ and TC6 for $R_{min} = 0.99$, in order to find the optimum configuration of α and β for the problem at hand. For both selected problem instances, the proposed ACO algorithm is applied using twenty-five possible combinations of α and β, with each parameter ranging between 1 and 5. To account for the heuristic nature of the ACO algorithm, the algorithm is run ten independent times at each of the twenty-five parameters' settings. Since this experiment involves a significantly large number of runs, we adopted the conventional two-mode SN model in all reliability calculations in this experiment. This significantly decreases the computational burden of calculating the fitness function expressed in (4.6) as can be inferred from the discussion in Section 3.6.2 in Chapter 3. Similar to the previous section, we use the conventional two-mode SN model in all reliability calculations to decrease the computational cost of this experiment without affecting its outcome.

Fig. 4.7 and Fig. 4.8 show the average value of the total number of deployment points, i.e. the average deployment cost, in the obtained solutions the versus α and β. Fig. 2 shows that there is an advantage in setting $\alpha = 1$ and $\beta = 3$, at which the minimum average deployment cost is obtained. Fig. 3 also shows that the minimum average deployment cost is obtained by setting $\alpha = 1$ and $\beta = 3$ in addition to setting $\alpha = 2$ and $\beta = 5$. Hence, in the

following experiments we set $\alpha = 1$ and $\beta = 3$. Table 4.7 shows the settings of the different parameters of the proposed ACO algorithm.

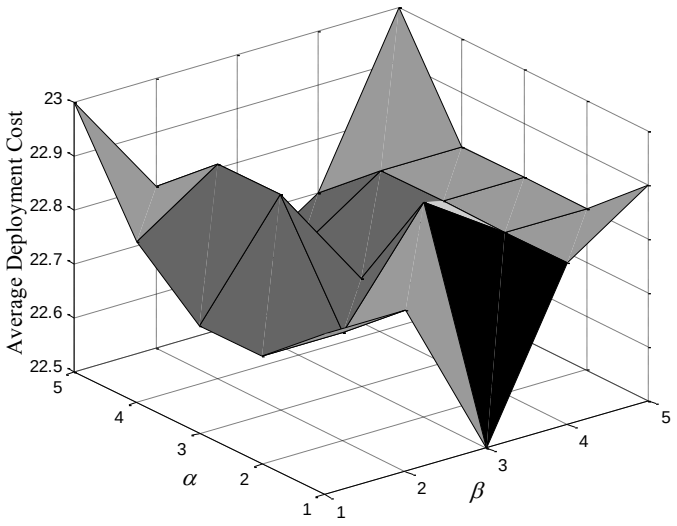


Fig. 4.7 The average deployment cost obtained from applying the proposed ACO algorithm on test case TC3 at $R_{min} = 0.9999$. The best combination of α and β is (1,3).

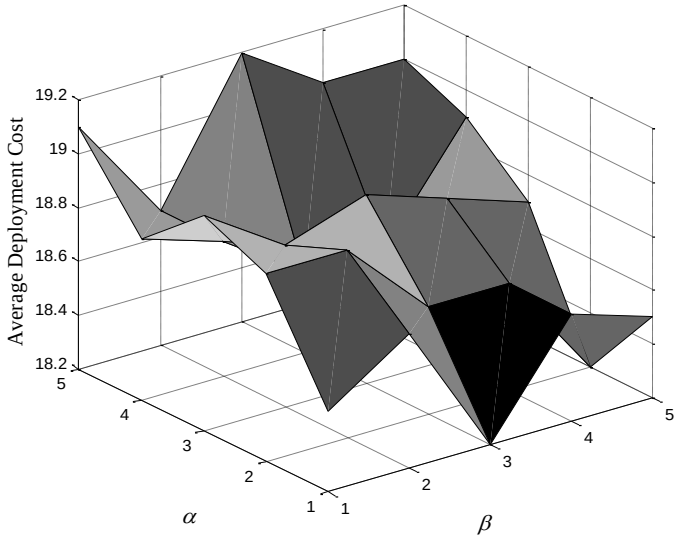


Fig. 4.8 The average deployment cost obtained from applying the proposed ACO algorithm on test case TC6 at $R_{min} = 0.99$. The best combinations of α and β are (1,3) and (2,5).

Table 4.7 Parameters of the proposed ACO algorithm

Parameter	Setting
Number of ant m	30
Pheromone influence parameter α	1
Heuristic info influence parameter β	3
Pheromone levels update rule	MMAS
Pheromone evaporation rate ρ	0.5
Pheromone constraining parameter $\mathscr{b}$	10
Heuristics	LS procedure in Table 4.4
Maximum No. of Generations n_{max}	100
No. of Generations for Convergence n_{conv}	20

4.4.3. A GH for Benchmarking the Proposed Algorithms

Since, to the best of our knowledge, the proposed MA and ACO algorithms are the first algorithms for solving the MCRC-SDP, a benchmark approach is required to evaluate their performance. Similar to the studies in [121], [94] and [73], we benchmark the performance of the proposed algorithms using a GH. The GH uses the same heuristic information adopted in the proposed ACO algorithm. It also follows the same basic idea of constructing solutions to the MCRC-SDP by consecutively building connected covers until the combined reliability of the connected covers meets or exceeds the specified minimum reliability R_{min}. The pseudo code of the GH is given in Table 4.8.

The input to the GH includes all the MCRC-SDP instance parameters: $\boldsymbol{D}, \boldsymbol{T}, d_0, R_{min}, r_s, r_c, \lambda_s, \lambda_t, \lambda_p$ and λ_b. The GH is initialized by an empty solution superset (i.e. $\boldsymbol{S} = \varphi$) and a connected cover index $k = 0$. The GH then proceeds in rounds. In each round, a deployment point is added to the connected cover with the current index k, denoted by $\boldsymbol{S}_k$. Similar to the proposed ACO algorithm, connectivity to the sink node located at d_0 is achieved by constricting the candidate deployment points for inclusion to $\boldsymbol{S}_k$ in a given round to the set $\boldsymbol{\mathcal{N}}_{next}$. Similar to the neighborhood set $\boldsymbol{\mathcal{N}}_i^a$ defined in Section 4.3.2.2 and expressed in (4.8) and (4.11), the definition of the set $\boldsymbol{\mathcal{N}}_{next}$ depends on whether the current round is an intra- or inter- connected cover round. For an intra-connected cover round, $\boldsymbol{\mathcal{N}}_{next}$ has a similar definition to the set $\boldsymbol{\mathcal{N}}_{ifull}^a$ expressed in (4.9), which includes all the deployment points which have not been added in previous rounds to the solution and are within the communication range r_c of any of the deployment points belonging to $\boldsymbol{S}_k$. For an inter-connected cover round (which includes the first round, i.e. the start of the first connected cover), on the other hand, $\boldsymbol{\mathcal{N}}_{next}$ has a similar definition to the set $\boldsymbol{\mathcal{N}}_{sink}^a$, which includes all the deployment points which have not been added in previous rounds to the solution and are within the communication range r_c of the sink node located at d_0. For both types of rounds, the GH calculates the coverage gain $\mathscr{g}_j$, as defined in Section 4.3.2.2, of all the deployment points $d_j \in \boldsymbol{\mathcal{N}}_{next}$ and adds the point with the *highest* gain, denoted by $\mathscr{g}_{max}$, to $\boldsymbol{S}_k$. In the case where more than one deployment point have the maximum gain or if none of the deployment points belonging to $\boldsymbol{\mathcal{N}}_{next}$ have a non-zero coverage gain, the GH chooses a deployment point from $\boldsymbol{\mathcal{N}}_{next}$ randomly. The GH terminates when the combined reliability of the constructed connected covers meets the reliability constraint, i.e. $R(\boldsymbol{S}) \geq R_{min}$.

Table 4.8 Pseudo code of the GH used for benchmarking the performance of the proposed algorithms for solving the MCRC-SDP

Procedure GREEDY_HEURISTIC

Input: $\boldsymbol{D}, \boldsymbol{T}, d_0, R_{min}, r_s, r_c, \lambda_s, \lambda_t, \lambda_p$ and λ_b
Initialize: $\boldsymbol{S} = \varphi$, $R(\boldsymbol{S}) = 0, k = 0, \boldsymbol{D}^- = \boldsymbol{D}$
While $R(\boldsymbol{S}) < R_{min}$
  Build a new connected cover: $k \leftarrow k + 1$, $\boldsymbol{S}_k = \varphi$, $\boldsymbol{T}_{cov} = \varphi$
  While $\boldsymbol{T}_{cov} \neq \boldsymbol{T}$ (i.e. $\boldsymbol{S}_k$ is not a complete connected cover)
    If $\boldsymbol{S}_k = \varphi$, i.e. the beginning of $\boldsymbol{S}_k$
      $\boldsymbol{\mathcal{N}}_{next} = \{d_j \in \boldsymbol{D}^-: \|d_j d_0\| \leq r_c\}$
    Else
      $\boldsymbol{\mathcal{N}}_{next} = \{d_j \in \boldsymbol{D}^-: \|d_j d_{j'}\| \leq r_c$ for any $d_{j'} \in \boldsymbol{S}_k\}$
    End If
    Calculate coverage gain $\mathcal{g}_j \forall d_j \in \boldsymbol{\mathcal{N}}_{next}$
    Update $\boldsymbol{S}_k$ by adding $d_j \in \boldsymbol{\mathcal{N}}_{next}$ with $\mathcal{g}_j = \mathcal{g}_{max}$
    Update $\boldsymbol{T}_{cov}$ (i.e. update coverage of $\boldsymbol{S}_k$)
  End While
  Update $\boldsymbol{S}$: $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \boldsymbol{S}_k$
  Calculate $R(\boldsymbol{S}_k)$ and Update $R(\boldsymbol{S})$
  Update $\boldsymbol{D}^-$: $\boldsymbol{D}^- \leftarrow \boldsymbol{D}^- - \boldsymbol{S}_k$
End While
Output: $\boldsymbol{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_N\}$, $R(\boldsymbol{S})$

4.4.4. Comparison and Discussion

To carry out our evaluations, we apply the three algorithms, MA, ACO and GH, to the twenty one MCRC-SDP problem instances generated from the seven test cases in Table 4.5 at a minimum required reliability $R_{min} = 0.99, 0.999$ and 0.9999. For each problem instance, ten independent runs of each of the three algorithms are performed. For the MA and ACO algorithm, the parameters settings listed in Tables 4.6 and 4.7, respectively, are used. The algorithms are run on a workstation with the following specs: an Intel Xeon processor, CPU E5620, 2.4 GHz and 24 GB RAM.

Tables 4.9, 4.10 and 4.11 summarize the results, in terms of the quality of solutions/deployment cost obtained by the three algorithms for the seven test cases at $R_{min} = 0.99, 0.999$ and 0.9999, respectively. The tables show the lowest ('Best'), highest ('Worst') and the average ('Avg.') deployment cost obtained from each algorithm. The tables also show the success rate ('SR') in percentage of each method in finding a solution to each MCRC-SDP instance that fulfills all the constraints of the problem, i.e. a feasible solution. For each problem instance, the best result(s)/lowest deployment cost obtained among the three algorithms is written in bold. To provide a statistically accurate comparison in terms of the averages of the obtained results, a set of pair-wise Wilcoxon Signed-Rank test [137] is performed to confirm the initial observations drawn from the results in Tables 4.9 – 4.11. Tables 4.12 – 4.14 show the resulting *p*-values of the performed statistical test and the corresponding conclusion for each problem instance. Values less than the specified statistical significance, α=0.01, indicate that the null hypodissertation of equal averages is rejected. The alternative hypodissertation of the test is therefore true with a 99% confidence level, where "A<B" means B performs better than A (the true mean of the obtained deployment cost from B is lower than from A), while "A≈B" means A and B perform almost similarly.

Table 4.9 Comparison among the GH, MA and ACO algorithms in terms of quality of the obtained solutions for the test cases in Table 4.5 at $R_{min} = 0.99$

Test Case	Greedy Heuristic				Proposed MA				Proposed ACO Algorithm			
	Best	Worst	Avg.	SR(%)	Best	Worst	Avg.	SR(%)	Best	Worst	Avg.	SR(%)
TC1	10	11	10.1	40	**8**	10	8.4	100	**8**	8	8	**100**
TC2	11	12	11.1	70	**8**	10	9.1	100	**8**	8	8	**100**
TC3	11	19	12.5	30	**10**	11	10.2	100	**10**	10	10	100
TC4	12	22	15.8	50	**10**	11	10.8	100	**10**	10	10	100
TC5	19	22	20.4	40	**18**	19	18.4	100	**18**	18	18	100
TC6	20	24	21.8	40	19	20	21.8	100	**18**	19	18.4	100
TC7	21	23	22.2	70	20	21	20.7	100	**19**	20	19.2	100

Table 4.10 Comparison among the GH, MA and ACO algorithms in terms of quality of the obtained solutions for the test cases in Table 4.5 at $R_{min} = 0.999$

Test Case	Greedy Heuristic				Proposed MA				Proposed ACO Algorithm			
	Best	Worst	Avg.	SR(%)	Best	Worst	Avg.	SR(%)	Best	Worst	Avg.	SR(%)
TC1	15	17	15.6	30	**13**	14	13.2	100	**13**	13	13	100
TC2	15	18	16.7	40	14	15	14.4	100	**13**	13	13	100
TC3	18	29	23.3	20	**16**	17	16.8	100	**16**	16	16	100
TC4	18	26	21.4	40	17	18	17.3	100	**16**	16	16	100
TC5	26	29	27.4	30	25	26	25.3	100	**24**	24	24	100
TC6	28	31	29.7	20	26	28	27.1	100	**25**	26	25.7	100
TC7	30	32	31	20	27	29	28.4	100	**26**	27	26.1	100

Table 4.11 Comparison among the GH, MA and ACO algorithms in terms of quality of the obtained solutions for the test cases in Table 4.5 at $R_{min} = 0.9999$

Test Case	Greedy Heuristic				Proposed MA				Proposed ACO Algorithm			
	Best	Worst	Avg.	SR(%)	Best	Worst	Avg.	SR(%)	Best	Worst	Avg.	SR(%)
TC1	21	22	21.8	20	**18**	19	18.3	100	**18**	18	18	100
TC2	22	26	23.5	10	20	22	20.8	100	**18**	18	18	100
TC3	27	38	34.3	0	23	24	23.7	100	**22**	23	22.6	100
TC4	24	34	30	30	23	25	23.8	100	**22**	22	22	100
TC5	33	37	35.7	10	31	34	32.8	100	**30**	31	30.8	100
TC6	36	40	37.7	10	34	35	34.2	100	**32**	33	32.4	100
TC7	38	42	38.9	10	34	36	35.4	100	**33**	34	33.5	100

Table 4.12 Results of the pairwise Wilcoxon signed-rank tests on the test cases at $R_{min} = 0.99$ at 99% confidence level

Test Case	GH vs. MA[1]	MA vs. ACO[2]	Conclusion
TC1	9.69×10^{-5}	3.88×10^{-2}	**GH < MA ≈ ACO**
TC2	2.93×10^{-5}	7.83×10^{-5}	**GH < MA < ACO**
TC3	1.11×10^{-4}	8.37×10^{-2}	**GH < MA ≈ ACO**
TC4	5.03×10^{-5}	2.20×10^{-4}	**GH < MA < ACO**
TC5	2.97×10^{-4}	1.68×10^{-2}	**GH < MA ≈ ACO**
TC6	1.42×10^{-4}	1.27×10^{-4}	**GH < MA < ACO**
TC7	1.28×10^{-4}	8.92×10^{-5}	**GH < MA < ACO**

Table 4.13 Results of the pairwise Wilcoxon signed-rank tests on the test cases at $R_{min} = 0.999$ at 99% confidence level

Test Case	GH vs. MA[1]	MA vs. ACO[2]	Conclusion
TC1	4.52×10^{-5}	8.37×10^{-2}	**GH < MA ≈ ACO**
TC2	3.06×10^{-4}	2.28×10^{-5}	**GH < MA < ACO**
TC3	5.43×10^{-5}	2.20×10^{-4}	**GH < MA < ACO**
TC4	2.09×10^{-4}	2.02×10^{-5}	**GH < MA < ACO**
TC5	1.37×10^{-4}	2.02×10^{-5}	**GH < MA < ACO**
TC6	8.03×10^{-5}	1.15×10^{-4}	**GH < MA < ACO**
TC7	6.04×10^{-5}	4.18×10^{-5}	**GH < MA < ACO**

Table 4.14 Results of the pairwise Wilcoxon signed-rank tests on the test cases at $R_{min} = 0.9999$ at 99% confidence level

Test Case	GH vs. MA[1]	MA vs. ACO[2]	Conclusion
TC1	3.68×10^{-5}	3.84×10^{-2}	**GH < MA ≈ ACO**
TC2	9.09×10^{-5}	2.38×10^{-5}	**GH < MA < ACO**
TC3	6.31×10^{-5}	4.82×10^{-4}	**GH < MA < ACO**
TC4	7.18×10^{-4}	2.38×10^{-5}	**GH < MA < ACO**
TC5	2.64×10^{-4}	1.36×10^{-4}	**GH < MA < ACO**
TC6	4.71×10^{-5}	4.09×10^{-5}	**GH < MA < ACO**
TC7	6.39×10^{-5}	2.20×10^{-4}	**GH < MA < ACO**

All tests are carried out at a 99 percent confidence interval, i.e. $\alpha = 0.01$
"A<B" means B performs better than A, i.e. the true mean of B is lower than A, while "A≈B" means A and B perform similarly, i.e. there is no sufficient evidence to reject the null hypodissertation of equal true means.

1. p −values of the Wilcoxon signed-rank test of the alternative hypodissertation that the mean of the proposed MA is lower than the mean of the GH.
2. p −values of the Wilcoxon signed-rank test of the alternative hypodissertation that the mean of the proposed ACO algorithm is lower than the mean of the proposed MA.

Results in Tables 4.9 - 4.11 demonstrate that the proposed MA and ACO algorithms are consistently capable of finding fully feasible solutions to the problem at hand with a success rate of 100% for all the twenty one problem instances under consideration. This proves that the underlying design of the different building blocks of the two proposed algorithms is indeed appropriate for the problem at hand. For the MA, the demonstrated ability to converge to feasible solutions to the MCRC-SDP with a success rate of 100% for all the tested problem instances is attributed to the design of the chromosome fitness function expressed in (4.6). As discussed in Section 4.3.1.2, the reliability and redundancy constraints of the MCRC-SDP expressed in (4.4) and (4.5) are not automatically met in each chromosome. This in turn means that only a part of the genotypic space corresponds to feasible solutions to the MCRC-SDP. However, both of these constraints are incorporated in the fitness function by means of the two penalty terms which dampen the fitness of the chromosomes that correspond to solutions which violate one or both constraints. As a result, the MA search is directed towards the regions in the genotypic space which correspond to fully feasible solutions. This process is accelerated by the application of the LS procedure which aims to increase the fitness of the chromosomes to which it is applied by attempting to remove the redundancy and reliability constraints' violations in the corresponding solutions. For the proposed ACO algorithm, the demonstrated ability to consistently find feasible solutions is attributed to the design of the cost function expressed in (4.14). Some ants may construct infeasible tours due

to the violation of the redundancy constraint if one or more of the connected covers in the corresponding solutions are non-minimal. However, the cost function penalizes these infeasible tours and consequently the pheromone trail levels will reinforce feasible tours and increase the proportion of ants which construct feasible solutions over that of ants which construct non-feasible ones. This process is accelerated by the use of the proposed LS procedure outlined in Table 4.3.

On the other hand, the success rate of the GH does not exceed 70% and is, on average, considerably lower than 70%. For all the problem instances under consideration, the GH was able to construct solutions that satisfy the constraints expressed in (4.2) - (4.4). However, it failed in a considerable number of runs in obtaining solutions that satisfy the redundancy constraint expressed in (4.5), i.e. solutions that consist *only* of minimal connected covers and hence are feasible solutions. The failure of the GH in consistently constructing feasible solutions to the MCRC-SDP problem which consist only of minimal connected covers can be attributed to its 'greedy' method it follows in constructing the connected covers. This method aims at reducing the chance of adding redundant deployment points to the connected covers but fails to eliminate it completely. A redundant deployment point or points can be added to a given connected cover $\boldsymbol{S}_k$ in the following case. At any given stage in constructing $\boldsymbol{S}_k$, all the deployment points belonging to the set $\boldsymbol{\mathcal{N}}_{next}$ may happen to have a zero coverage gain. In this case, the GH selects a deployment point at random from $\boldsymbol{\mathcal{N}}_{next}$ to maintain the connectivity of $\boldsymbol{S}_k$. Hence, the selected deployment point is redundant to $\boldsymbol{S}_k$ in terms of coverage but non-redundant in terms connectivity thus far. However, depending on the deployment points selected by the GH in the following rounds till the completion of $\boldsymbol{S}_k$, this redundant deployment point(s) in terms of coverage may become redundant in terms of connectivity as well, meaning that its elimination from $\boldsymbol{S}_k$ would not compromise its coverage or connectivity. Hence, such a deployment point(s) becomes fully redundant and consequently $\boldsymbol{S}_k$ becomes a non-minimal connected cover.

It can also be observed from the GH results shown in Tables 4.9 - 4.11 that for each of the seven test cases, the success rate of the GH declines as the value of R_{min} increases. This behaviour is expected since the number of connected covers required to satisfy the reliability constraint increases with the increase of the value of R_{min}. As the number of connected covers the GH has to construct to meet R_{min} increases, the probability that a non-minimal connected cover is constructed increases as well. Consequently, this increases the probability that the GH obtains a non-feasible solution with one or more non-minimal connected covers which constitutes a failure.

Examining the results in Tables 4.9 - 4.11 also reveals that the two proposed algorithms significantly outperform the GH in terms of the deployment cost in all tested problem instances. For the proposed MA, in all instances, the algorithm obtained a minimum deployment cost (i.e. 'Best' solution) solution which is lower than that obtained by the GH. The difference between the averages of both algorithms is also significant in favour of the proposed MA. For the proposed ACO, results relative to those of the GH are even better. In all the problem instances, the highest deployment cost (i.e. 'Worst' solution) obtained by the ACO algorithm is lower than the lowest total number of deployment points in the solutions (i.e. 'Best' solution) obtained by the GH. This implies that even for the problem instances where the GH succeeded in obtaining solutions consisting only of minimal covers (i.e. feasible solutions) with a success rate higher than null, the proposed MA and ACO algorithms were capable of finding feasible solutions with significantly higher quality, i.e. solutions of a lower deployment cost. These observations are confirmed by the statistical tests carried out and summarized in Tables 4.12 - 4.14. These tests conclude that for all the tested problem instances, the two proposed algorithms outperform the GH with a confidence level of 99%. This proves that the two algorithms are appropriately designed and tuned for the

MCRC-SDP. It is also worth mentioning that for the majority of the test problem instances, the MA and ACO algorithm solutions consistently have a higher combined reliability levels than those of the feasible GH solutions. This is because the reliability of a minimal connected cover is inversely proportional to its size, provided that it has no coverage redundancy. The reliability in this case is equal to the probability of the existence of a single event that all the deployed SNs are in the *on* state. This probability increases when there are fewer deployed SNs, i.e. when there are fewer deployment points in the minimal connected cover. Therefore for the used problem instances, where the feasible solutions obtained from the three algorithms consist of minimal connected covers with no coverage redundancy, the lower the deployment cost, the higher the combined reliability level of the connected covers.

Results of test cases TC3 and TC4 show a greater advantage of the proposed MA and ACO algorithm over the GH in terms of solution quality as compared to the rest of the test cases at the three considered levels for R_{min}. This can be explained as follows. In these problem instances, some of the GH obtained solutions (both feasible and partially feasible) consisted of an entire additional connected cover when compared to the solutions obtained by the MA and ACO algorithms. This is because in these problem instances the GH constructed one or more minimal connected covers of non-optimal size (i.e. with lower reliability) in its earlier rounds, i.e. at the beginning of constructing a solution to the problem. This has caused the GH to have to construct an additional connected cover to meet the reliability constraint. This situation does not occur in the solutions obtained by the proposed MA and ACO due to their search efficiency.

As far as the comparison between the MA and ACO algorithms is concerned, the results in Tables (4.9) - (4.11) suggest that the ACO algorithm outperforms the MA in the majority of the tested problem instances. In terms of the average of the obtained solutions, this conclusion is further confirmed by the results of the statistical tests performed and which are presented in Tables (4.12) - (4.14). In terms of the lowest deployment cost (i.e. 'Best' solution), the MA exhibits a similar performance to the ACO algorithm in almost half of the tested problem instances. However, the MA's performance in that aspect deteriorates with the increase of R_{min}, i.e. deteriorates with the increase of the difficulty level of the problem. Furthermore, the MA results show a higher level of variation, i.e. a higher standard deviation, than the results obtained by the ACO algorithm. This behaviour can be attributed to the fact that the framework of the ACO is better suited to the MCRC-SDP. Specifically, the design of the ACO tour construction ensures that for every ant tour, the reliability constraint expressed in (4.4) is met by the constructed solution. Also, the neighbourhood definitions expressed in (4.8) - (4.12) ensure that for every ant tour, the level of SN redundancy in each constructed connected covers is non-existent at best or very low at worst. Hence, especially as the ACO algorithm progresses and the pheromone levels reinforce the feasible solutions, a significant portion of the ants in each iteration build feasible solutions. Furthermore, for those ants that build infeasible solutions, the LS procedure may convert that infeasible solution to a feasible one if the reliability constraint can still be met after pruning.

On the other hand, the chromosome encoding scheme in the proposed MA cannot guarantee that the reliability or the redundancy constraints are met in each chromosome. Therefore, only a fraction of the genotypic space correspond to feasible solutions. This in turn decreases the efficiency of the search performed by the MA in comparison with the ACO algorithm, especially with the increase on R_{min}. As R_{min} increases it becomes a more difficult task to find high quality feasible solutions and a higher chance that the MA will converge to a local minimum. This conclusion is supported by the following pattern in the results shown in Tables 4.9 - 4.11. At $R_{min} = 0.99$, the MA was able to find the minimum deployment cost (i.e. 'Best' solution) in five out of the seven test cases. At $R_{min} = 0.999$ it

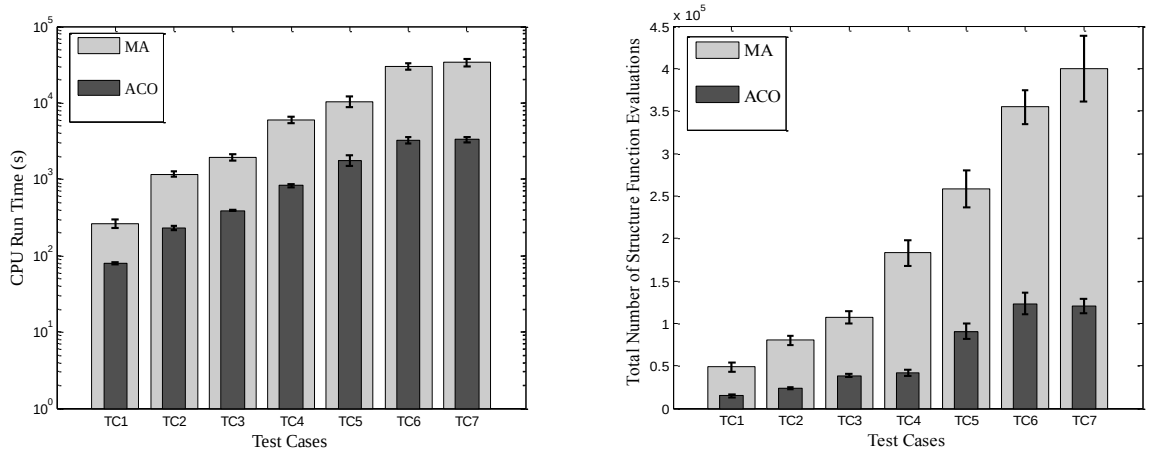
was able to do that in only two test cases. At $R_{min} = 0.9999$ it was able to do that in only one test case.

It can also be observed from Tables 4.9 - 4.11 that the solutions obtained by the MA have a significantly higher variability, i.e. a higher standard deviation, than the solutions obtained from the ACO algorithm. This behaviour can be explained as follows. According to the ant tour construction procedure, all ants start their tours at the sink node location in the RoI, which is generated randomly for every test case. Their choice of the first deployment point to add to every constructed connected cover, including the first one, is stochastic since it is governed by the probabilistic transition rule expressed in (4.7). However, these deployment points are constricted by the feasible neighborhood $\boldsymbol{\mathcal{N}}_i^a$ expressed in (4.11). On the other hand, the solution obtained by the MA in each run is highly affected by the fitness of the chromosomes in the initial generation/population which is randomly selected. Hence, the results obtained from the ACO algorithm have a lower sensitivity to the random initialization which originates from the transition rule in (4.7) than the sensitivity of the results obtained by the MA to the random initialization of its first population.

Fig. 4.9(a), 4.10 (a) and 4.11(a) show a comparison between the MA and ACO algorithm in terms of the computational cost, measured by the CPU run-time in seconds, for $R_{min} = 0.99$, 0.999 and 0.9999, respectively. Fig. 4.9(b), 4.10 (b) and 4.11(b) show a comparison between the MA and ACO algorithm in terms of the total number of performed structure function evaluations in the reliability calculations of the fitness and cost functions, for $R_{min} = 0.99$, 0.999 and 0.9999 respectively. The 95% confidence interval is indicated in both sets of figures.

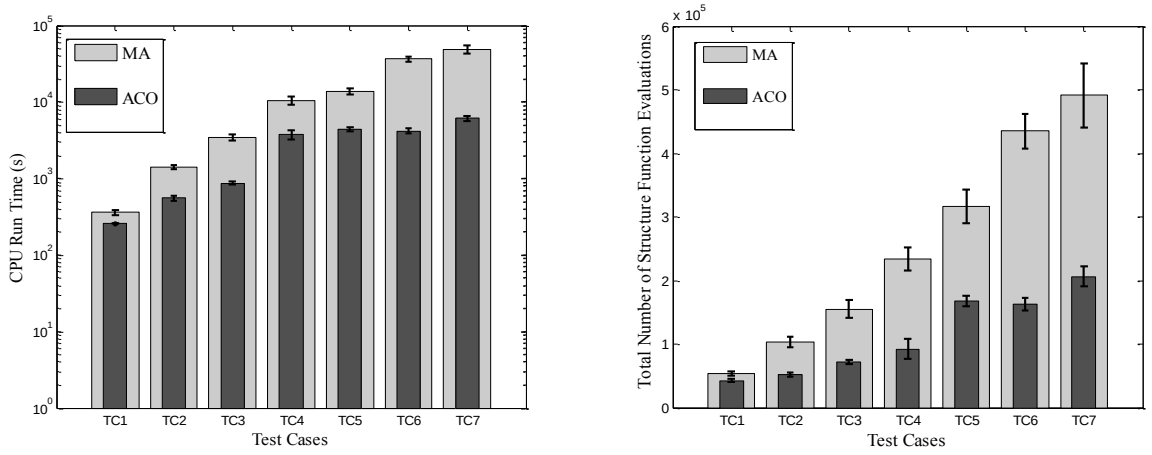
Examining Fig. 4.9(a), 4.10 (a) and 4.11(a) shows that for each test case and for both the MA and the ACO algorithm, the increase in the value of R_{min} causes an increase in the computational cost, i.e. the run-time of the algorithm. This is an expected behavior since increasing R_{min} raises the difficulty level of the problem as solutions are expected to contain a higher number of minimal connected covers. Examining Fig. 4.9(b), 4.10 (b) and 4.11(b) confirms this conclusion. For each test case, as R_{min} increases, the average number of total performed structure function evaluations increases significantly.

Results illustrated in Fig. 4.9, 4.10 and 4.11 also show that the computational cost of the two algorithms is affected differently by the increase in the problem scale. For the proposed



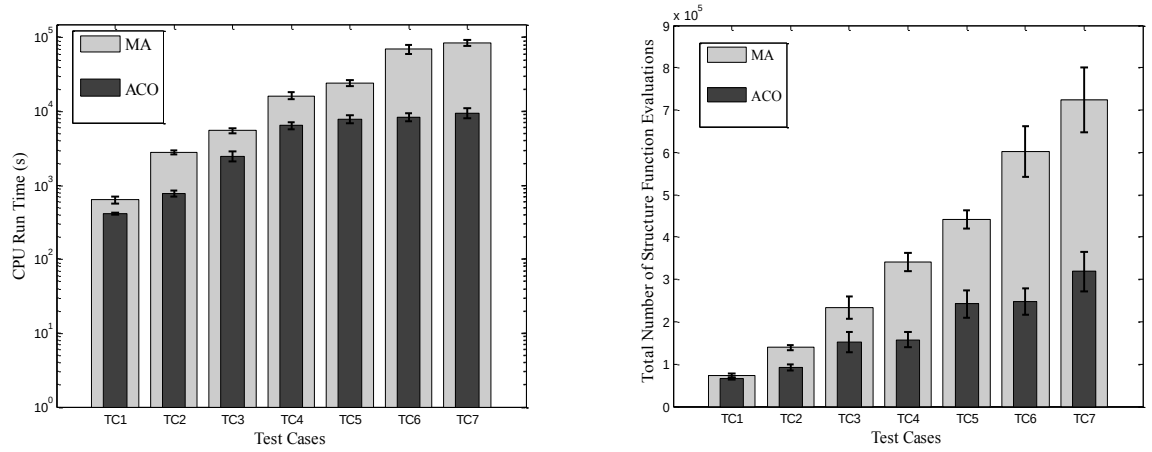
(a) (b)

Fig. 4.9 Computational cost of the proposed MA and ACO algorithm for test cases in Table 4.5 at $R_{min} = 0.99$ measured using (a) CPU run-time in seconds, (b) Total number of performed network structure function evaluations.



(a) (b)

Fig. 4.10 Computational cost of the proposed MA and ACO algorithm for test cases in Table 4.5 at $R_{min} = 0.999$ measured using (a) CPU run-time in seconds, (b) Total number of performed network structure function evaluations.



(a) (b)

Fig. 4.11 Computational cost of the proposed MA and ACO algorithm for test cases in Table 4.5 at $R_{min} = 0.9999$ measured using (a) CPU run-time in seconds, (b) Total number of performed network structure function evaluations.

MA, increasing the value of $|\boldsymbol{D}|$ consistently increases the computational cost of the algorithm. This can be attributed to the chromosome encoding scheme in which the chromosome length is equal to $|\boldsymbol{D}|$ to preserve the one-to-one correspondence between the chromosome genes and the deployment points in a given test case. The size of the population is also directly proportional to the problem scale as $\mu = 2|\boldsymbol{D}|$. For the proposed ACO algorithm, the correlation between $|\boldsymbol{D}|$ and the computational cost is less drastic compared to the MA. This is primarily because the number of ants m remains unchanged for all the test cases under consideration.

Results in Fig. 4.9, 4.10 and 4.1 also show that for all the tested instances, the ACO algorithm has a significantly lower computational cost than the MA. This behavior can be attributed to two main factors. The first factor is the higher population size of the MA compared to the number of ants in the ACO algorithm. The second factor is the fact the execution time of a structure function evaluation directly proportional to the size of the connected cover for which the evaluation is carried out. Due to the chromosome encoding scheme used in the MA, chromosomes may represent connected covers with high levels of redundancy, especially in the earlier iterations of the algorithm. On the other hand, the ant

tour construction procedure and the feasible neighborhood definitions are designed to minimize the chance of ants constructing non minimal connected covers. Hence, during the execution of both algorithms, the average size of the connected covers the MA calculates reliability for is higher than that in the case of the ACO algorithm. This in turn means that the average execution time of a structure function evaluation is higher in the case of the MA than the ACO algorithm.

4.5. Chapter Summary

In this chapter, we considered the problem of deploying a WSN that meets a specified minimum level of reliability during its mission time at a minimum network deployment cost. To minimize the internal interference, bandwidth usage and energy consumption throughout the network's mission time, we defined the problem as the problem of finding a number of non-overlapping minimal connected covers of the targeted region of interest such that the combined reliability level of these connected covers meets or exceeds the specified minimum level of reliability while the deployment cost is kept at a minimum. We coined this problem the MCRC-SDP and proved that it is NP-complete. To measure the reliability of the network, we used the reliability metric proposed in Section 3.5.3 in Chapter 3. We proposed two stochastic optimization algorithms to solve the defined MCRC-SDP, namely, a MA (i.e. HGA) and an ACO each coupled with a LS procedure. For each of the proposed algorithms, we discussed the design of the different building blocks of the algorithm. To benchmark the performance of both algorithms in terms of the quality of the obtained solutions, we presented a GH which attempts to find good quality solutions for the MCRC-SDP by using the same underlying idea of building solutions and heuristic information adopted in the proposed ACO algorithm. Finally, we presented and discussed our experimental results of applying the three algorithms to twenty one MCRC-SDP problem instances with different scales and required minimum reliability levels. Experimental results showed that for all the tested problem instances, the proposed MA and ACO algorithm significantly outperform the GH in terms of the quality of the obtained solutions, which validates the design of both algorithms. Experimental results also showed that the ACO algorithm outperforms the MA in the majority of the tested problem instances in terms of quality/optimality and in all the tested problem instances in terms of the computational cost. Results also suggest that the proposed ACO algorithm exhibits better scalability to the dimensions of the MCRC-SDP (determined by $|\boldsymbol{D}|$ and $|\boldsymbol{T}|$) than the proposed MA.

Chapter 5

A Practical Realization of the Proposed Reliable Cost-Optimal Deployment Technique

5.1. Introduction

Each of the minimal connected covers belonging to the solution of the MCRC-SDP problem provides complete coverage of the target locations in RoI and forms a connected network with the sink node. According to the formulation of the MCRC-SDP presented in Section 4.2.2 in Chapter 4, these minimal connected covers are assumed to be activated in an orthogonal manner. This assumption is adopted to minimize the level of internal interference in the network during its mission time and is fundamental to the formulation of the reliability constraint expressed in (4.5). There is therefore a need to coordinate the sleep and active cycles of the SNs belonging to these connected covers based on the dynamic state of the overall deployment throughout the network mission time. This type of SN activity coordination is known in the WSN literature as *topology control*.

The goal of a topology control protocol designed for a reliable WSN deployment obtained from solving the MCRC-SDP defined by (4.1) - (4.5) is to activate the deployed minimal connected covers in the assumed orthogonal manner as follows. A single minimal connected cover is activated at any given point in time during the network mission time while the SNs belonging to the remaining connected covers are put in sleep mode. This activated minimal connected cover remains active until its functionality is compromised due to random failures of one or more of its constituent SNs. At that point, the remaining functional SNs belonging to this minimal connected cover are put in sleep mode till the end of the network mission time and another minimal connected cover is activated. This procedure is repeated until either the mission time of the network elapses or there are no remaining functional deployed minimal connected covers. According to the statement of the MCRC-SDP, the probability of the first event is equal to R_{min} and that of the second event is equal to $1 - R_{min}$.

In this chapter, we propose a topology control protocol for reliable cost-optimal WSN deployments obtained from solving the MCRC-SDP using the proposed algorithms presented in Chapter 4. We implement the proposed protocol on a WSN simulator and apply the proposed protocol on different deployment scenarios. We present and discuss the experimental results in terms of two protocol performance metrics: the incurred overhead and the time required to detect and repair the functionality of the WSN due to potential SN failures. The different factors which affect each of these performance metrics are also highlighted.

5.2. Previous Work on WSN Topology Control

In WSN literature, the term *topology control protocol* encompasses any protocol that controls some aspect of the deployed SNs in a WSN deployment (e.g. SN state (sleep/active), transmission range (radio power level), mobility (if the WSN contains mobile SNs)) to affect the network's level of coverage and/or connectivity [138]. In the context of problem at hand, we will focus on the topology control protocols which manage the states of the deployed SNs

in WSNs, which are termed *Temporal* Topology Control Protocols (TTCP). The concept of temporal topology control is based on the premise that WSNs, either deployed in a planned or a random fashion, can have a certain level of SN redundancy, i.e. the number of deployed SNs in the network may greatly exceed the number required to form a single connected-cover of the RoI. To manage this redundancy, a TTCP pushes a subset of the deployed SNs in the WSN into sleep mode while keeping the other SNs active to fulfil the coverage and connectivity requirements of the network in an adaptive fashion [139]. An SN in sleep mode will conserve its energy expenditure by turning off its sensor(s) and most or all of its transceiver circuitry (depending on whether the SN will be activated by an internal timer or an external WAKEUP message). Hence, it will not be able to sense the environment or communicate wirelessly. As a direct consequence, TTCPs can enhance the WSN energy-efficiency and hence prolong the network lifetime (in case of a random deployment). TTCPs are also used to counteract internal interference within the WSN, specifically those adopting a contention-based medium access control scheme (e.g. non-beacon enabled mode of the IEEE 802.15.4). Internal interference occurs mainly due to packet collisions, which in turn increases with the increase of the number of active SNs in the network. By minimizing the number of active SNs in the network at any given time without compromising the network functionality in terms of coverage and connectivity, a TTCP decreases packet collisions by decreasing the amount of redundant traffic. This results in increasing the overall WSN throughput [36], [100].

TTCPs can be classified according to *where* they are executed in the network, i.e. can be classified as either *centralized* or *distributed* protocols [140]. A centralized TTCP is executed in the sink node(s) of the WSN and requires global information on the SNs such as their relative location from the sink node and their current state and mode (i.e. in sleep or active mode, functional or failed state) at any time instant. Distributed TTCP is executed concurrently on the SNs and required only local information (i.e. information obtained from the neighbors of each SN). Topology control protocols can also be classified according to *how* they are executed, *iteratively* or *non-iteratively*. The general approach of an iterative TTCP is to *periodically* compute a *single* connected-cover of the RoI from the pool of functional deployed SNs. In other words, the protocol divides time into equal periods or *rounds*. At the beginning of each round, the protocol, either in a centralized or distributed fashion, is responsible for activating only a subset of the deployed SNs such that this subset forms a connected-cover of the RoI. After the current round elapses, this procedure is repeated, where possibly a different subset of SNs is activated, based on the states of the deployed SNs at the beginning of the new round. This scheme continues until the mission time of the WSN has elapsed or until there are not enough functional SNs left to construct a connected-cover. On the other hand, the general approach adopted by a non-iterative TTCP is to *pre*-divide the deployed SNs into non-overlapping connected-covers before actually controlling their sleep/active cycles. Consequently, the original topology control problem of minimizing the number of active SNs is transformed into an optimization problem with the objective of maximizing the number of the non-overlapping connected covers. A given connected-cover is said to be *active* when all the SNs belonging to it are put in active mode. The aim of this approach is to apply an orthogonal sleep-wake scheme on the obtained non-overlapping connected covers.

Examples of iterative TTCPs are presented in [36], [141] - [149]. In [36], the authors propose the Optimal Geographical Density Control (OGDC) protocol. OGDC protocol runs periodically, i.e. in *rounds*, in a distributed fashion. The objective of OGDC is to activate the minimum number of SNs in each round such that the selected active SNs completely cover the targeted RoI. The protocol is based on the assumptions that all deployed SNs are homogeneous, have a binary disk coverage profile and a communication range that is equal to

or exceeds twice the sensing range. This latter assumption is used to prove that connectivity of WSN during any round is maintained if the targeted RoI is fully covered. The protocol assumes that each SN can have three states: *on*, *off* or *undecided*. In the beginning of each round, SNs go through the *decision phase* of the protocol, during which all the functional deployed SNs assigned to an *undecided* state. A SN in the *undecided* state has its transceiver in a fully functional state, i.e. it can send and receive messages. Then the process of selecting the active SNs proceeds by randomly assigning one or more of the SNs to be "starting SNs". These starting SNs change their state to the *on* state and broadcast "power-on" messages to the network. A SN in the *on* state will remain active throughout the current round after the decision phase elapses. Each SN still in the *undecided* state utilizes the information provided by the "power-on" messages it receives to carry out local computations to decide whether it change its state to the *on* or *off* state or remain *undecided* to wait for more "power-on" messages. These local computations factor in the relative location of current *on* SNs, the residual energy of the SN, the coverage gain it will contribute (if any) if it becomes active. This process continues until all SNs are either in the *on* or *off* states, which marks the end of the decision phase of the protocol. Simulation results show that OGDC outperforms existing protocols presented in [146] and [147] in terms of the average percentage of active SNs in a given round and provides almost the same level of coverage as the best protocol. Results also show that applying OGDC to a given deployment results in a longer lifetime of the network when compared to the protocol in [147].

In [141], the authors present a distributed token-based iterative TTCP named Coverage-Centric Active Nodes Selection (CCANS). Similar to OGDC in [36], CCANS runs in rounds and each round has a protocol decision phase. During the protocol decision phase, a SN can assume three states, namely *active*, *sleep* or *unset*. However, throughout the decision phase of CCANS all SNs have their transceivers in a fully functional state regardless of their state. At the beginning of the decision phase of each round, all SNs are in the *unset* state and they are required to construct a list of 1-hop neighbor SNs using "hello" messages. The proposed CCANS protocol is then executed in two separate stages. In both stages the protocol assigns the token to a single SN at any time instant during its execution and that SN is called the *token node*. Stage 1 is given the name *sensing coverage evaluation*. In this stage, the token node performs a set of coverage calculations based on its local information to decide a temporary change from its *unset* state to either an *active* or *sleep* states. This change is temporary because the state decision is taken while some of the other SNs are in the *unset* state and hence the state of a given SN may be changed again in stage 2 of the protocol. The current token node then sends a "state" message to its neighbors and passes the token to one of its neighbors based on specific criteria. The steps are repeated until all SNs are either in *active* or *sleep* states. This marks the conclusion of stage 1 and initiates stage 2 of the protocol. Stage 2 is given the name *node state and connectivity checking* and is only executed by SNs assigned to a *sleep* state during stage 1. Using the same token approach as in stage 1, a token node repeats its coverage calculations with no neighbor SNs now in the *unset* state. It also checks if any of its active neighbor SNs is not connected to the sink node. Based on these two steps, the token node decides its *final* state for the current round sends an "update" message to its neighbors and passes the token on. Stage 2 is completed when all SNs finalize their state decision (either *active* or *sleep* throughout the current round), which marks the end of the decision phase. Simulation results show that CCANS protocol increases the average percentage of sleeping SNs in a given round when compared to similar existing protocols presented in [142] and [148].

In [143], the authors argue that an iterative TTCP designed for *p*-percent coverage (PPC) instead of full coverage provides the network operator with the ability to dynamically specify and tune the required level of network coverage, specifically for applications where

extending the network lifetime is the primary objective and not reliability. Based on this argument, the authors propose a distributed iterative TTCP named Distributed $p-$Percent Coverage Protocol (DPCP). Each round of DPCP consists of three stages, namely *discover*, *construct* and *connect*. Each stage has a fixed time interval. It is assumed that at the beginning of each round, all SNs are in the *active* state, and that they remain active throughout the execution time of the three stages of DPCP. In the *discover* stage, each functional deployed SN discovers its neighbors using 1-hop "hello" messages. In the *construct* stage, a subset of SNs sufficient to achieve PPC decide to join the *active set* for the current round depending on a set of local coverage calculations and 1-hop message exchanges with neighbors. In the *connect* stage, possibly more SNs decide to join the *active set* for the current round to ensure its connectivity by the means of both broadcast messages (sent by the SNs belonging to the *active set* thus far) and more local calculations. After DPCP concludes, all SNs not belonging to the active set finalized in the last stage are put to sleep until a new round begins. A test bed using Java is built to simulate the operation of DPCP. Simulation results show that that DPCP outperforms the Greedy Selection (GS) protocol presented in [149] in prolonging the lifetime of the network.

Similar to [143], the study in [144] proposes an iterative distributed TTCP designed for *p*-percent coverage (PPC). However, the authors in [144] assume that the deployed SNs can have different sensing and communication ranges, i.e. the WSN is heterogeneous and that the SNs are deployed in a uniform (i.e. random) dense deployment in the RoI. They also define the term *redundancy degree* of a SN, which is defined as the percentage of overlap in coverage area between the SN and its first-hop neighbors of different types. Based on this definition and the stated assumptions, the authors derive an analytical expression for redundancy degree of a SN, which depends on the SN type, the number of the SN first-hop neighbors and their types. This analytical expression is used to compute a *redundancy table* for each SN type, which provides the redundancy degree for different combinations of first-hop neighbors' numbers and types. The authors assume that this table is either stored in the SNs off-line or sent to the SNs by the sink node after deployment as a control message. In the proposed TTCP, SNs can assume three states: *active*, *wait* and *sleep*. At the beginning of each round of the protocol, all SNs are in the *active* state and they broadcast "hello" messages to the network. Each SN uses the "hello" messages it received to construct a *neighbor table* which contains the ID, type and current state of its first-hop neighbors (the table is updated using periodic "hello" messages during the decision phase of each round). Based on the required percentage of coverage, each SN decides whether it's potentially redundant in this round or not using the redundancy and neighbor tables. If it is potentially redundant, it changes its state to the *wait* state and invokes a random back-off timer. If the SN is still redundant at the end of the back-off time (i.e. the SN still has sufficient neighbors in the *active* or *wait* states), it sends a "sleep" message to its neighbors and switches to the *sleep* state till the end of the current round. Otherwise it switches to the *active* state. Simulation experiments are carried out using ns 2.34. Simulation results shows that the proposed protocol is more capable than the protocol proposed in [145] in prolonging the lifetime of the network since it requires a smaller average number of active SNs.

On the other hand, examples of non-iterative TTCPs are presented in [150] - [152], [121]. In [150], the authors consider the problem of maximizing the number of the non-overlapping covers in a dense randomly deployed WSN. They coin the problem the Disjoint Sets Cover (DSC) problem. The authors assume that for a set to be functional, only the complete coverage of a given set of target points in a specified RoI is required. Connectivity was not considered in this study. The authors assume that an orthogonal sleep-wake scheme is applied to the non-overlapping covers such that only one cover is activated at any point in time while the other covers are put in sleep mode. This cover remains active for

a fixed amount of time, after which another cover is activated in a round robin fashion. The authors argue that following this scheme will prolong the SNs' battery lifetime and minimize internal interference in the network. They proceed to solve the DSC problem by first proving that it is NP-complete. They then transform the DSC problem into a Maximum-Flow problem (MFP), which is then formulated and solved as a Mixed-Integer-Programming (MIP) problem. Finally, using the output of the formulated MIP, a heuristic named Maximum Covers using MIP (MC-MIP) is used to carry out the actual partitioning of the deployed SNs into the required non-overlapping covers. They propose that a central node (i.e. sink node) performs the computation of the non-overlapping covers, i.e. solves the DSC, once shortly after the network deployment. The authors assume that the deployed SNs have the ability to localize themselves and that they would send their location to the central node to enable it to carry out the computation of the non-overlapping covers. The central node would then send each SN its membership information so each SN can compute its own sleep-wake schedule (assuming SNs are time-synchronized using on-board GPS receivers or through periodic beacon messages). Presented results show that the proposed MC-MIP heuristic outperforms the heuristic proposed in [151] in terms of the number of non-overlapping covers obtained in various instances of the DSC problem.

Similar to [150] and [151], the authors in [152] also consider the problem of maximizing the number of the non-overlapping covers in a dense randomly deployed WSN, i.e. Disjoint Sets Cover (DSC) problem. As in [150], the authors assume that an orthogonal sleep-wake scheme is applied to the non-overlapping covers with the difference being that an activated cover remains activated until its coverage of the target locations in the RoI is compromised. No other details on how the non-overlapping covers are to be managed are provided. The authors critique the heuristic algorithm MC-MIP proposed in [150]. They point out that MC-MIP involves solving a MIP problem using an implicit exhaustive search which, although enhances the optimality of the MC-MIP heuristic, requires a computation time which in the worst case increases exponentially with the number of deployed SNs and hence is impractical to use for medium to large WSN layouts. Based on their critique, they propose a GA called GA for Maximum Disjoint Sets Cover (GAMDSC) as a more scalable alternative to MC-MIP and provide analysis of the time complexity of the algorithm in the worst case. To evaluate the performance of GAMDSC in terms of optimality (measured by the average number of obtained sets) in comparison to MC-MIP, the authors apply both algorithms on several WSN layouts with different number of deployed SNs but for the same set of target points. Results show that GAMDSC has a comparable performance to that of MC-MIP, with a small advantage for MC-MIP. However, GAMDSC offers significant saving in computation time compared to MC-MIP.

In [121], the authors consider a variation of the DSC problem in [150] - [152]. They assume that a number of sink nodes are deployed in the WSN and that the deployed SNs can only communicate with sink nodes. Consequently, the DSC problem is converted into the problem of maximizing the number of the non-overlapping *connected* covers in a dense randomly deployed WSN, where each connected cover is composed of a subset of the deployed SNs and a subset of the deployed sink nodes. Each connected cover must fulfil three constraints: namely the *coverage* constraint, the *collection* constraint and the *routing* constraint. The coverage constraint is simply that the SNs belonging to the connected cover provide full coverage of the RoI. The collection constraint is that all the SNs belonging to the connected cover must be able to communicate directly (1-hop) with at least one of the sink nodes belonging to the same connected cover. The collection constraint is that the sink nodes belonging to the same connected cover must form a connected network with each other. The authors propose an ACO algorithm named ACO-MNCC to solve the above variation of the DSC problem. Since the proposed approach is the first algorithm proposed for solving this

variation of the DSC problem, the authors also propose a greedy algorithm that uses the same heuristic information as MDS-MCC to compare ACO-MNCC against. Experimental results show that ACO-MNCC consistently outperforms the greedy algorithm for all tested sizes of the WSN. No details were provided in [121] on how an orthogonal sleep-wake scheme should be applied to the non-overlapping connected covers obtained by the proposed ACO-MNCC algorithm.

5.3. Proposed Topology Control Protocol for Reliable WSN Deployments

From the literature survey presented in Section 5.2, it is clear that iterative TTCPs are applied to WSNs that are characterized by a *dense random* deployment to increase the lifetime of such WSNs. This is done by activating only a subset of the functioning deployed SNs which form a connected cover of the RoI during each time round. However, iterative TTCPs also introduce a significant amount of SN processing and control traffic overhead during their decision phase(s) at the beginning of each round. This is because each SN must carry out local computations to decide its state and exchange this with its neighbors for state decision making purposes. This implies an energy cost on SNs during the decision phase(s) of each round [147]. This also implies an increase in the internal interference in the network as well. The shorter the duration of the round, the higher the incurred control traffic overhead and added internal interference that is experienced by the network. Increasing the duration of the rounds to decrease these undesired effects of the iterative TTCPs is not a practical solution since it decreases the network responsiveness to potential SN failures. This is because if one or more SN failures occur during a given round, the functionality of the WSN would remain compromised until the beginning of the next one. For example, the study in[140] proposes a round duration of 1000 seconds. In the situation where the reliability of WSN operation is a primary concern, such relatively large durations are not acceptable.

On the other hand, a TTCP based on a non-iterative approach can be applied easily to reliable cost-optimal deployments obtained from solving the MCRC-SDP. In the context of the defined MCRC-SDP, a reliable WSN deployment represented by $\boldsymbol{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_N\}$ consists of a number of non-overlapping *minimal* connected covers of the targeted RoI under the assumption that these minimal connected covers are activated orthogonally. Hence, the main objective of a non-iterative TTCP designed for $\boldsymbol{S}$ is to control the orthogonal activation of the minimal connected covers that comprises $\boldsymbol{S}$ with the lowest possible additional control overhead.

We will coin the proposed non-iterative TTCP the Reliable Deployment TTCP (RD-TTCP). Table 5.1 shows the pseudocode of the RD-TTCP functionality executed by the sink node while Table 5.2 shows the pseudocode of the RD-TTCP functionality executed by the SNs. The main idea of the RD-TTCP can be summarized as follows. The protocol runs periodically, i.e. the mission time of the network is divided into equal time intervals of duration t_p seconds. Similar to the studies in [141] - [149], we assume that SNs are time-synchronized. The information about the specific SNs in the N connected covers of $\boldsymbol{S}$ and the order of connected covers' activation is pre-programmed in the sink node. We will assume that the order of activation is based on the reliability of the connected covers in a descending order. In the beginning of each interval, i.e. at $t = nt_p, n = 0, 1, 2, \ldots$, all deployed SNs of all functional connected-covers self-activate and remain in the active state for the duration of a *listening period* of duration t_l seconds, where $t_l < t_p$. This means that SNs during the listening period of each interval can exchange packets with each other and with the sink node. At the beginning of the mission time of the network, i.e. at $t = 0$, the sink node checks the stored membership information on the deployed connected-covers and sends an "on"

Table 5.1 Pseudocode of the proposed RD-TTCP functionality executed by the sink node

RD-TTCP: SINK_NODE

Input: $\mathcal{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_N\}$ **,** timer values t_p , t_l, t_h , t_{ack}, network mission time T_m
Initialize: $\boldsymbol{S}_{on} = \emptyset$, connected-cover counter $k = 1$, network time $t = 0$
While $\{t < T_m \;\&\&\; k \leq N\}$
  If $\{t = nt_p, n = 0,1,2, \ldots \;\&\&\; \boldsymbol{S}_{on} = \emptyset\}$ // at the beginning of each interval when a new connected cover needs to be activated
    Send "on" msgs to all SNs $s_i \in \boldsymbol{S}_k$, $i \in [1,2, \ldots, |\boldsymbol{S}_k|]$; Set ack. timer to t_{ack}
    If { ACK received from $s_i \in \boldsymbol{S}_k \; \forall \; i = 1,2.., |\boldsymbol{S}_k| =$ TRUE before ack. timer expires }
      $\boldsymbol{S}_{on} = \boldsymbol{S}_k$; Set HB#$[j] = 0$, $j = 1,2,.., |\boldsymbol{S}_{on}|$ // current active set is $\boldsymbol{S}_k$
      Send "off" msgs to remaining functional SNs $s_i \in \boldsymbol{S}_{k-1}$, $i \in [1,2, \ldots, |\boldsymbol{S}_{k-1}|]$
    Else // the connected cover $\boldsymbol{S}_k$ being activated is compromised
      $k = k + 1$ // activate the next connected cover in the order of activation in the next listening period
    End If
  If $\{\boldsymbol{S}_{on} \neq \emptyset\}$ // if there is a functional activated connected cover
    While { $|max(\text{HB\#}) - min(\text{HB\#})| < 4$**}** //while current connected cover is still functional
      Listen: "hear-beat" msgs received from $s_j \in \boldsymbol{S}_{on} \rightarrow$ HB#$[j] \equiv$ "heart-beat" msg index of s_j
    End While
    $\boldsymbol{S}_{on} = \emptyset$ // current connected cover $\boldsymbol{S}_{on}$ has failed
    $k = k + 1$ // activate the next connected cover in the order of activation in the next listening period
  End If
End While
If $\{k > N\}$ // If there are no more functional connected covers in the deployment
  Output: "Network Failed"
Else
  Output: "Network Mission Time Elapsed"
End If
End RD-TTCP: SINK_NODE

message to the SNs which belong to the first connected cover in the order of activation. Based on this step, if an SN receives an "on" message from the sink node stamped with its own ID as the destination, it replies with an ACK message after a short random back-off timewithin the time interval t_{ack}. It will keep its active status until it receives an "off" message from the sink node at a later time. The short random back-off time associated with the ACK message is used to decrease the probability of packet collision due to several SNs sending ACK messages to the sink node after being activated by their "on" messages. At the end of the listening period, i.e. at $t = nt_p + t_l$, the sink node will decide whether the connected cover it attempted to activate is indeed activated. It bases its decision on whether it received an ACK message from every SN belonging to this connected cover. On the other hand, if an SN does not receive an "on" message during the listening period of a given interval, it switches itself (i.e. transceivers and sensors) off and maintains that state until its internal timer signals the start of the listening period of the next interval.

We will denote the current active connected cover at any given time during the mission time of the network by $\boldsymbol{S}_{on}$. As a result of the above steps, only the SNs of the first connected cover in the order of activation, denoted $\boldsymbol{S}_1$, remain active after the elapse of the listening time of the first interval, i.e. $\boldsymbol{S}_{on} = \boldsymbol{S}_1$. For the sink node to be able to monitor the functionality/health of the currently activated connected cover $\boldsymbol{S}_{on}$, SNs which belong to $\boldsymbol{S}_{on}$ will periodically send a "heart-beat" message to the sink node every t_h seconds directly following the end of the listening period. To minimize packet collisions, each SN of $\boldsymbol{S}_{on}$ will send its "hear-beat" message at a random time during each t_h interval, i.e. the sink node will

Table 5.2 Pseudocode of the proposed RD-TTCP functionality executed by the SNs

RD-TTCP: SENSOR_NODE

1 **Input:** timer values t_p , t_l, t_h , t_{ack}, network mission time T_m
2 **Initialize:** $state = off$, act_flag = FALSE , $deact\_flag$ = FALSE, network time $t = 0$
3 **While** { $t < T_m$}
4 **Process I** // if SN is neither activated or deactivated
5 **If** { ($nt_p \leq t \leq nt_p + t_l$, $n = 0,1,2,\dots$) && act_flag = FALSE && $deact\_flag$ = FALSE}
6 **Listen:** $state = on$ // turn sensor and transceiver on
7 If { "on" msg received from sink node with local ID = TRUE}
8 act_flag = TRUE ; record activation round index n_a; set "hear-beat" msg index HB#←1
9 Send ACK msg to sink node with a random back back-off time within t_{ack}
10 **Break & GO TO Process II**
11 **Else**
12 $state = off$ // SN goes to sleep at the end of the listening period if it was not activated
13 **End If**
14 **End If**
15 **Process II** // begins when the SN is activated by an "on" msg from the sink node during round n_a
16 **While** {act_flag = TRUE && $deact\_flag$ = FALSE}
17 Set "heart-beat" timer to t_h at $t = n_a t_p + t_l$
18 Send "heart-beat" msg with index HB# to sink node with a random back-off before "heart-beat" timer expires
19 **If** {"heart-beat" timer expires = TRUE && $deact\_flag$ = FALSE}
20 Reset "heart-beat" timer to t_h ; HB# ← HB# + 1
21 **GOTO line 18**
22 **End If**
23 **If** {"off" msg received from sink node with local ID = TRUE} // SN is now part of a compromised connected cover that is being set to sleep mode
24 $deact\_flag$ = TRUE ; $state = off$
25 **End If**
26 **End While** // end Process II since SN is now deactivated
27 **End While** // end Process I and II because network mission time T_m has elapsed
28 **End RD-TTCP: SENSOR_NODE**

receive a "heart-beat" message from each functioning SN once every t_h seconds. Since all the deployed connected covers are minimal, i.e. contain no completely redundant SNs, the total failure of any of the SNs which belong to $\boldsymbol{S}_{on}$ will compromise its functionality in terms of coverage or/and connectivity. Therefore, if the sink node does not receive three consecutive "heart-beat" messages from one or more of the SNs of $\boldsymbol{S}_{on}$, it will assume that a SN has failed and the next connected cover in the order of activation needs to be activated while the remaining functional SNs in the current $\boldsymbol{S}_{on}$ needs to be put to sleep. Hence, the sink node looks up the next connected cover in its list of the activation order, i.e. $\boldsymbol{S}_2$, and sends an "on" message to the SNs belonging to it in during the listening period of the next interval. After the sink node receives confirmation that the SNs which belong to $\boldsymbol{S}_2$ have received their "on" messages via the ACK messages sent by the SNs, the sink node sends an "off" message to the remaining functional SNs in $\boldsymbol{S}_{on}$ and changes the index pointer of $\boldsymbol{S}_{on}$ to the newly activated connected cover, i.e. $\boldsymbol{S}_{on} = \boldsymbol{S}_2$. The above procedure continues until the mission time of the network elapses or there are no more functional connected covers to activate. Fig. 5.1 illustrates the RD-TTCP functionality executed by both the sink node and the SNs as detailed above in the form of a state diagram.

5.4. Experimental Results and Discussion

In this section, we present the experimental results obtained from implementing the proposed RD-TTCP on the wireless sensor network simulator COOJA [153], [154]. COOJA

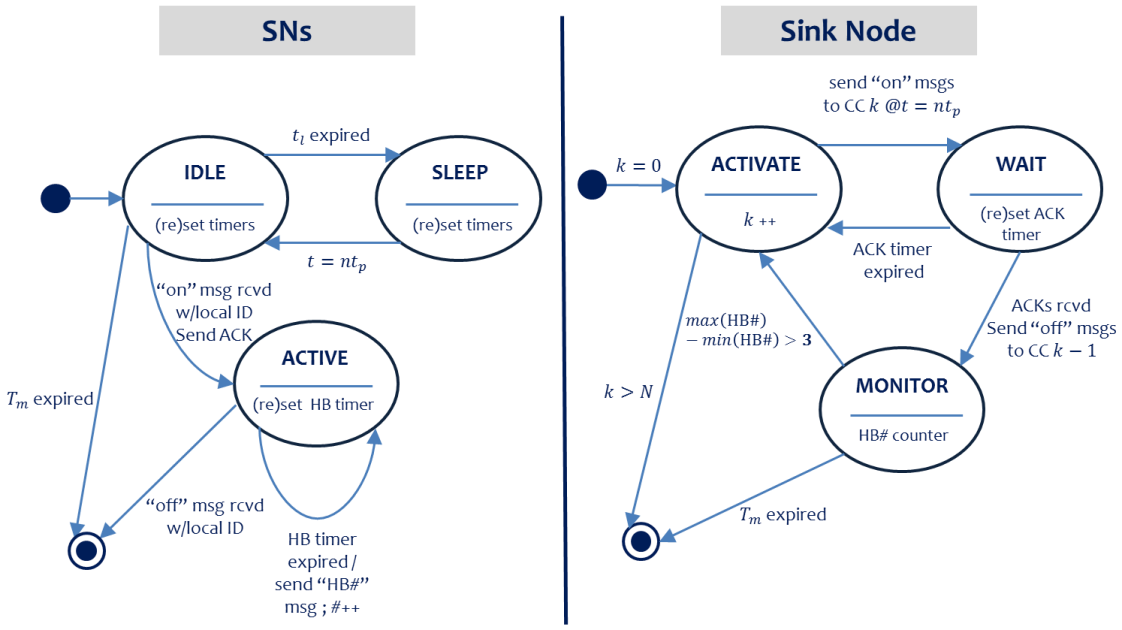


Fig. 5.1 A state diagram describing the proposed RD-TTCP functionality

is a flexible Java-based simulator designed for simulating networks of SNs running the Contiki operating system [155], [156]. COOJA is capable of emulating a wide variety of commercial wireless sensor nodes. The main advantage in using COOJA to simulate protocols proposed for WSNs is that it enforces the hardware limitations of the selected SN type (including memory, number of internal timers, speed of processing, etc.). Moreover, the developed code to be executed by each type of SN in a simulated WSN in COOJA is the exact same code that can be uploaded to a physical SN of the same type. Hence, using COOJA to develop and simulate new protocols for WSNs running the Contiki operating system enables a precise/realistic inspection and debugging of the developed code before implementing the code/protocol on physical SNs [157].

To implement the proposed RD-TTCP, two custom mote types are developed, namely, a sink node and a sensor node, each capable of performing the proposed protocol steps outlined in Tables 5.1 and 5.2 respectively. For both mote types, the Tmote sky hardware platform is selected. Tmote sky is a commercial ultra-low power IEEE 802.15.4 compliant (non-beacon enabled/contention based MAC) wireless sensor module [158]. The underlying routing protocol used in this experiment is the IPv6 Routing Protocol for Low Power and Lossy Networks, which is known in the literature as RPL. RPL is a distance-vector routing protocol designed by the Routing over Low-power and Lossy networks (ROLL) Working Group in order to cater to the specific needs of low-power and lossy networks such as WSNs designed for different IoT applications. It is specified in the Internet Engineering Task Force (IETF) standards document RFC 6550 [159]. It is worth mentioning that the implementation of the proposed RD-TTCP is independent of the underlying routing protocol.

To evaluate the performance of the proposed RD-TTCP, we test the protocol on several network scenarios. Each scenario represents a different reliable cost-optimal deployment. Specifically, we use a reliable cost-optimal deployment for each of the seven test cases presented in Table 4.5 at the minimum required reliability $R_{min} = 0.99$ obtained by the proposed ACO algorithm (i.e. we use one of the obtained solutions characterized by the best quality/lowest deployment cost) as summarized in Table 4.9. Table 5.3 summarizes the data pertinent to these deployments, where each is used to set up a simulation scenario in COOJA. The RoI is a two-dimensional square $100 \times 100\ m^2$ area. The location of the sink node d_0 in

the RoI is different for each test case as it was generated randomly as explained in Section 4.4.1 in Chapter 4.

Two performance metrics are considered: the communications overhead incurred by the proposed RD-TTCP and the time it takes the protocol to repair the WSN after an SN failure occurs, i.e. Time To Repair (TTR). The overhead is measured as the percentage of the RD-TTCP traffic (measured in bytes) out of the total RPL control traffic within the network. In this experiment, we assume that there is no traffic generated from the deployed SNs to convey sensory data to the sink node and hence the sources of traffic in the network are the proposed protocol and the RPL routing protocol. The second performance metric is measured using two time variables. The first time variable is the total time it takes the RD-TTCP to activate a new functional connected cover (i.e. to repair the WSN), measured from the instant the functionality of the current connected cover is compromised by a SN failure, which we will denote by TTR1. The second time variable is the time it takes RD-TTCP to activate a new functional connected cover (i.e. to repair the WSN), measured from the instant the sink node discovers that the functionality of the current connected cover is compromised, which we will denote by TTR2. For each simulation scenario, ten independent runs are by the simulator. Simulation and protocol parameters are summarized in Table 5.4.

Fig. 5.1 shows the proposed protocol communication overhead for each of the simulated deployments. For each deployment, the 95% confidence interval of the corresponding set of runs is indicated. Fig. 5.1 suggests that the proposed protocol have a low overhead, with averages that do not exceed 3.5% for all the tested deployments. The relatively low overhead of the proposed protocol is due to its simplicity and the fact that only the SNs which belong to the currently activated connected cover communicate with the sink node through sending the periodic heartbeats. SNs belonging to non-activated and failed connected covers do not generate any protocol traffic. Fig. 5.1 also shows a large variation in the measured protocol overhead, indicated by the wide confidence intervals. This can be attributed to the fact that the amount of both the RD-TTCP traffic and the RPL control traffic are highly sensitive to changes in the topology of the WSN due to the SN failure events during the course of the simulation. An SN failure event can either trigger a repair action in the network (i.e. the activation of a new connected cover) or not. The first case occurs when the failed SN belongs to the currently activated connected cover and hence its functionality as well as the network's functionality is compromised. The latter case occurs when the failed SN belongs to one of the connected covers that have not been activated yet while the currently activated connected cover remains functional.

The lowest measured values of the proposed protocol overhead occurred in simulation runs in which there were no SN failure events or in which the SN failure events do not trigger a repair action. In these simulation runs the amount of traffic generated by the RD-TTCP is at its lowest since only one connected cover is activated in the beginning of the simulation and only the SNs which belong to that connected cover communicate periodically with the sink node by sending their "heart-beat" messages. On the other hand, the amount of control traffic generated by the RPL is at its highest since all or most of the deployed SNs are functional until the end of the simulation time. This in turn means that all or most of the deployed SNs generate routing control traffic during the simulation time. For all simulated deployments, the RD-TTCP overhead for these simulation runs is roughly 1%.

On the other hand, the higher measured values of overhead occurred in simulation runs in which SN failure events triggered repair actions. In these cases, the RD-TTCP traffic increases since additional protocol messages are generated, namely, the activation messages required to activate a new connected cover and the "off" messages sent to the remaining functional SNs in the failed connected cover. On the other hand, the RPL control traffic

Table 5.3 Data of the reliable cost-optimal deployments used to create the COOJA simulation scenarios

Test Case	Total number of SNs ($\|\boldsymbol{\mathcal{S}}\|$)	Number of connected covers (N)	Connected Covers Distribution
TC1	8	2	{4,4}
TC2	8	2	{4,4}
TC3	10	2	{4,6}
TC4	10	2	{5,5}
TC5	18	3	{6,6,6}
TC6	18	3	{6,6,6}
TC7	18	3	{6,6,6}

Table 5.4 Simulation and Protocol Parameters

Parameter	**Setting**
Simulation time	30 minutes
SN hardware platform	Tmote Sky
PHY + MAC	802.15.4 non-beacon enabled
Routing Protocol	IPv6 RPL
SN failure probability λ	0.08
SN failure time distribution	uniform
Protocol round duration t_p	120 seconds
Protocol listening period t_l	30 seconds
Protocol heart-beat message frequency t_h	30 seconds

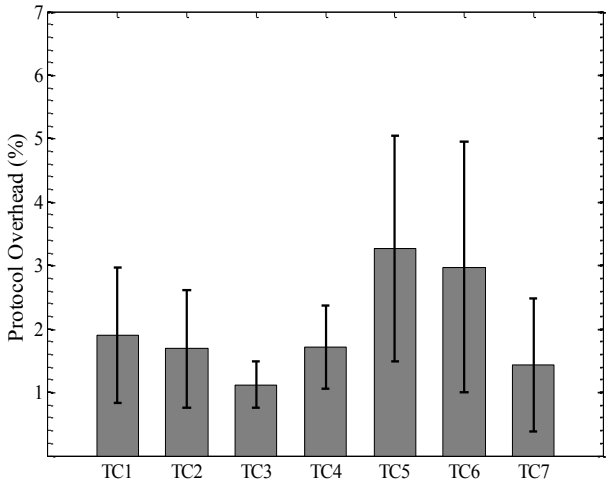


Fig. 5.2 Traffic overhead incurred by the proposed RD-TTCP, measured as ratio between RD-TTCP and routing traffic in percentage points.

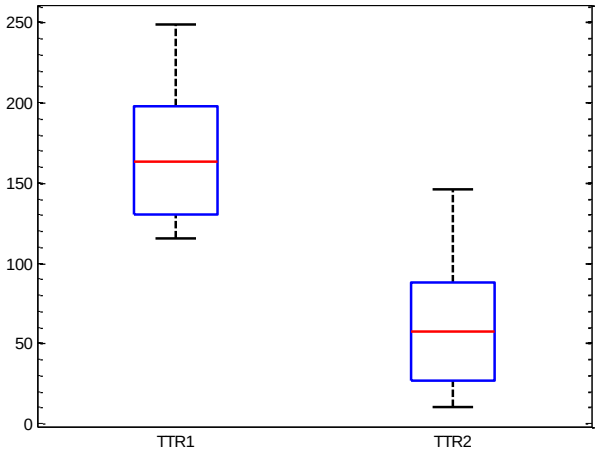


Fig. 5.3 Boxplots of the proposed RD-TTCP time to repair metric, measure by the time variables TTR1 and TTR2

decreases significantly since in these cases several SNs which belong to the failed connected cover are forced to sleep. Hence, these SNs stop generating any RPL control traffic until the end of the simulation run. It follows that the time at which the repair action is taken during the simulation run affects the value of the protocol overhead: the earlier the repair action is taken, the lower the total RPL control traffic generated during the simulation run and hence the higher the overhead. However, Fig. 5.1 shows that the upper confidence interval of the proposed protocol overhead does not exceed 5% for all the simulated deployments which means that even when one or more repair action is carried out during the network mission time, the overhead incurred by the proposed protocol remains relatively low.

Fig. 5.2 shows the box plots for the TTR time variables, TTR1 and TTR2, for all the SN failure events that triggered a repair action/event in the seventy simulation runs performed (10 runs for each of the seven deployments). The time variable TTR2 represents the response time of the proposed protocol to an identified SN failure which compromises the functionality of the current connected cover. The average response time (i.e. the average value of TTR2) is primarily dependent on the protocol interval duration t_p. This is because once the sink node discovers that a SN failure that requires a repair action has occurred, it needs to wait until all the SNs which belong to the other functional (yet not yet activated) connected covers emerge from sleep mode at the beginning of the listening period of the next protocol interval. Fig. 5.2 indicates that the median of TTR2 is equal to 57 seconds which is an expected value since in this experiment t_p is set to 120 seconds. Values of TTR2 higher than the set value of t_p (i.e. higher than 120 seconds) can occur in cases where the next connected cover in the order of activation has already suffered a complete SN failure and hence it functionality is compromised. In this case, the proposed protocol will require an additional protocol interval to discover that the functionality of the connected cover it is trying to activate it compromised through not receiving an ACK message from one or more of the member SNs. This event, however, has a low probability of occurrence as its probability is only half the probability that two consecutive connected covers fail (the higher the number of connected covers in the deployment, the lower that probability will be for a given SN failure probability). It should be pointed out that although decreasing the protocol round duration can decrease the average response time of the proposed protocol, it can also increase the RPL routing traffic since it will increase the frequency at which the RPL

structure is disrupted (through SNs alternating between sleep and active modes at a higher frequency). Hence, the setting of the protocol round duration t_p requires careful tuning according to the type of routing protocol used in the WSN to keep the routing overhead at a reasonable level.

On the other hand, the relationship between time variables TTR1 and TTR2 indicates the speed at which the sink node discovers/identifies SN failures in the currently activated connected cover. This speed is primarily dependent on two factors. The first factor is the frequency at which "heart-beat" messages are sent, which is set by the timer value t_h. The second factor is the number of the "heart-beat" messages the sink node has to miss from one or more of the SNs in the active connected cover before it decides that the connected cover has failed. In this experiment, t_h is set to 30 seconds and the number of missed "heart-beat" messages is set to 3. Accordingly, the minimum amount of time required for the sink node to discover a SN failure in the active connected cover is three times the value of t_h or 90 seconds. Fig. 5.2 indicates that the difference between the medians of TTR1 and TTR2 is 106 seconds which is an expected value for the selected values of the protocol parameters. It should be pointed out that configuring the number of missed "heart-beat" messages to take a lower value may decrease the TTR1 and hence the speed of network repair. However, this may also lead to misdiagnosing, unnecessary repairs and an unnecessary increase in overhead especially if the WSN is deployed in lossy environments where the link quality and hence the Packet Delivery Ratio (PDR) is low. Hence, the setting of the protocol parameter requires careful tuning according to the expected link quality and the conditions of the environment in which the WSN is to be deployed.

It should be noted that a comparision with one or more of the existing iterative TTCPs proposed in [36], [141] – [145], [147] , [149] was not possible. This because the objective of these protocols as explained earlier is to prolong the lifetime of dense randomly-deployed WSN. This is in contrast with the proposed RD-TTCP which designed to manage the minimal connected covers in a relaible cost-optimal planned WSN deployment. As such, the performance parameters for iterative TTCPs and the proposed non-iterative RD-TTCP are different. For example, in [36], [141] , [142], [144], [145], [147] and [149] the primary performance metric is the percentage of the sleeping nodes to the total number of nodes in the WSN after the execution of the protocol in a single round. Additional performance metrics such as the ratio of the RoI area actually covered to the total area, the lifetime of the network and rate of SN energy consumption were evaluated in [36], [142] - [145] and [149]. The only study that commented on the overhead of the iterative TTCP is [147], where overhead was calculated as the average percentage of the energy spent on protocol messages to the total energy expenditure of SNs. However, the results provided in [147] can not be used as a benchmark since many of the simulation parameters and assumptions are different, primarily the fact that in [147], SNs are assumed to periodically send data messages whereas in our simulation no data traffic was generated.

5.5. Chapter Summary

In this chapter, we introduced a practical realization of the reliable deployment algorithms that we proposed earlier. For this purpose, we defined the concept of temporal topology control in WSNs as a method for controlling the sleep/active cycles of the SNs deployed in a WSN. We highlighted the importance of TTCPs, specifically in minimizing excessive traffic in the WSN, which in turn reduces packet collisions. We discussed why existing TTCPs are not suitable for WSN designed for critical applications. Accordingly, we proposed a TTCP suitable for managing the disjoint connected covers which constitute the

reliable cost-optimal WSN deployments obtained by solving the MCRC-SDP. The proposed non-iterative TTCP was coined the RD-TTCP. The proposed RD-TTCP was implemented on the WSN simulator COOJA and applied to several different reliable cost-optimal WSN deployments. Simulation results suggest that the overhead incurred by the proposed protocol and the average TTR are relatively low and hence the proposed protocol is applicable in practice.

Chapter 6

Conclusions

6.1. Dissertation Summary

In this dissertation, we presented our research contributions on the topic of reliable cost-optimal deployment of static WSNs. We defined the dissertation problem as the problem of deploying a static WSN that meets an application-specific reliability level at a minimum network deployment cost. We coined this problem the Minimum-Cost Reliability-Constrained SN Deployment Problem (MCRC-SDP). We highlighted the significance of this problem to the body of research on WSN deployment. Based on the problem definition, we identified the different research objectives associated with it. We then presented the work carried out to address each of the identified research objectives.

We presented a survey and classification of the existing WSN deployment algorithms based on their underlying mathematical approach. The presented classification contains four major mathematical approaches: Genetic Algorithms (GAs), Computational Geometry (CG), Artificial Potential Fields (APFs) and Swarm Intelligence (SI). We discussed the strengths and limitations of the four approaches in terms of different WSN design factors. We concluded that GAs and Ant Colony Optimization (ACO) are the most suitable approaches for deploying static WSNs with multiple design objectives. We presented an experimental comparison among three of the existing WSN deployment algorithms based on these two approaches. Their performance was evaluated in terms of optimality, speed of convergence and scalability.

We then identified the key SN related and non-SN related issues that affect the reliability of a WSN. We discussed the existing WSN reliability metrics in the literature. Based on this discussion, we proposed a novel WSN reliability metric to address some of the limitations of the existing metrics. The proposed metric adopts practical assumptions concerning the operation and configuration of the WSN. It also adopts a more accurate SN model compared to the simplistic model adopted in the existing metrics. A search algorithm is presented to calculate the proposed metric in a computationally efficient manner. Extensive experimental results on the proposed metric are presented and discussed. The experimental results demonstrated the computational efficiency of the developed search algorithm. They also showed that the proposed metric has a significantly higher accuracy in measuring WSN reliability compared to the most-relevant existing metric in the literature.

Based on this metric, we presented the mathematical formulation of the dissertation problem, i.e. the MCRC-SDP. The MCRC-SDP is formulated as a constrained combinatorial optimization problem which we prove to be NP-Complete. We proposed two heuristic optimization algorithms that are designed to find high quality solutions for the MCRC-SDP. The first algorithm is a Memetic Algorithm (MA), which is also known in the literature as a Hybrid GA (HGA). The second algorithm is an ACO algorithm coupled with a Local Search (LS) heuristic. For each algorithm, the design of the different basic building blocks is discussed. Extensive experimental results are presented, analyzed and discussed to highlight the strengths and limitations of each algorithm. Results show that the ACO algorithm outperforms the MA in the majority of the tested problem instances in terms of optimality and in all the tested problem instances in terms of the computational cost.

Finally, we reviewed some of the significant studies presented in the literature on the topic of WSN topology control. We discussed why existing WSN Topology Control Protocols (TCPs) are not suitable for managing the reliable cost-optimal deployments obtained from solving the MCRC-SDP. Accordingly, we proposed a practical TCP that is suitable for managing the non-overlapping minimal connected covers which constitute such deployments. We presented experimental results obtained from implementing the proposed TCP and applying it to several deployments. Results suggest that the overhead incurred by the proposed protocol is relatively low and hence is the proposed protocol is applicable in practice as a realization of our optimized deployment techniques.

6.2. Future Work

The work presented in this dissertation can be further extended in the following directions:

- Other heuristic optimization techniques can be applied to solve the MCRC-SDP. For example, Differential Evolution (DE) [160] (which is an evolutionary algorithm that shares the same foundations as the GAs) and Bee Colony Optimization (BCO) [161] (which has recently been introduced as a new direction in SI) are good candidates.
- The proposed MA and ACO algorithm can be extended to allow for the deployment of heterogeneous reliable cost-optimal WSNs (i.e. WSNs composed of different types of SNs with varying capabilities).
- The MCRC-SDP can be extended to be a multi-objective constrained optimization problem by considering other design objectives such as minimizing routing cost.
- An original GH capable of consistently finding feasible solutions to the MCRC-SDP can be designed.
- The SN probabilities of failure due to other non-SN related issues such as attacks on the deployed SNs by external malicious forces can be included/considered in the WSN reliability metric.
- The performance of the proposed TCP for managing the MCRC deployments can be implemented and studied in conjunction with other routing protocols such as Adhoc-On Demand Distance Vector (AODV) [162].

References

[1] P. Rawat, K. D. Singh, H. Chaouchi and J. M. Bonnin, “Wireless sensor networks: a survey on recent developments and potential synergies,” *Journal of Supercomputing*, vol. 68, no. 1, pp. 1-48, 2014.

[2] L. Fagen and P. Xiong, "Practical secure communication for integrating wireless sensor networks into the Internet of Things," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3677-3684, 2013.

[3] A. Flammini and E. Sisinni, “Wireless sensor networking in the Internet of Things and cloud computing era,” *Proc. of the European Conference on Solid-State Transducers (EUROSENSOR 2014)*, vol. 87, pp. 672-679, 2014.

[4] P. Harrop and R. Das, “Wireless Sensor Networks (WSN) 2014-2024: Forecasts, Technologies, Players,” IDTechEx, Cambridge, United Kingdom, Tech. Rep, pp. 1.1-1.10, 2014.

[5] I. Stoianov, L. Nachman, S. Madden, T. Tokmouline, and M. Csail, “Pipenet: A wireless sensor network for pipeline monitoring,” *in Proc. of 6th International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 264-273, 2007.

[6] L. Lei, Y. Kuang, X. S. Shen, K. Yang, J. Qiao and Z. Zhong, "Optimal reliability in energy harvesting industrial wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 8, pp. 5399-5413, Aug. 2016.

[7] J. Chinrungrueng, U. Sununtachaikul and S. Triamlumlerd, “A vehicular monitoring system with power-efficient wireless sensor networks,” *in Proc. of International Conference on ITS Telecommunications*, pp. 951-954, 2006.

[8] W. Xue, L. Wang and D. Wang, "A prototype integrated monitoring system for pavement and traffic based on an embedded sensing network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1380-1390, 2015.

[9] A. Milenkovic, C. Otto and E. Jovanov, “Wireless sensor networks for personal health monitoring: Issues and an implementation,” *ElSevier Computer Communications*, vol. 29, no. 13, pp. 2521-2533, 2006.

[10] A. Sawand, S. Djahel, Z. Zonghua Zhang and F. Nait-Abdesselam, "Multidisciplinary approaches to achieving efficient and trustworthy eHealth monitoring systems," *IEEE/CIC International Conference on Communications in China (ICCC)*, pp.187-192, 2014.

[11] M. Naderan, M. Dehghan and H. Pedram, “Mobile object tracking techniques in wireless sensor networks*,” in Proc. of International Conference on Ultra-Modern Telecommunications (ICUMT)*, 2009.

[12] V. Jelicic, T. Razov, D. Oletic, M. Kuri and V. Bilas, “Maslinet: A wireless sensor network based environmental monitoring system,” *in Proc. of the International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 150-155, 2011.

[13] Y. Liao et al., "SnowFort: An open source wireless sensor network for data Analytics in infrastructure and environmental monitoring," *IEEE Sensors Journal*, vol. 14, no. 12, pp. 4253-4263, 2014.

[14] M. Ishizuka and M. Aida, “Performance study of node placement in sensor networks,” *in Proc. 24th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pp. 598-603, 2004.

[15] S. Ghatak, S. Bose and S. Roy, "Intelligent wall mounted wireless fencing system using wireless sensor actuator network," *Proc. of International Conference on Computer Communication and Informatics (ICCCI)*, 2014.

[16] E. Kantoch, P. Augustyniak, M. Markiewicz and D. Prusak, "Monitoring activities of daily living based on wearable wireless body sensor network," *Proc. of IEEE International Conference on Engineering in Medicine and Biology Society (EMBC)*, pp. 586-589, 2014.

[17] Ping Wang, Yan Yan, Gui Yun Tian, Omar Bouzid and Zhiguo Ding, "Investigation of wireless sensor networks for structural health monitoring," *Journal of Sensors*, vol. 2012, Article ID 156329, 2012.

[18] M. Sha, D. Gunatilaka, C. Wu and C. Lu, "Empirical study and enhancements of industrial wireless sensor-actuator network protocols," *IEEE Internet of Things*, early-access article, 2017.

[19] W. Kuo and M. J. Zuo, *Optimal reliability modeling: principles and applications*, John Wiley & Sons, 1st edition, 2003, pp. 85-102.

[20] F. Koushanfar, M. Potkonjak and A. Sangiovanni-Vecentelli, "Fault-tolerance in wireless sensor networks", *in Handbook of Sensor Networks*, CRC Press, 1st edition, 2005.

[21] D. Deif and Y. Gadallah," Classification of wireless sensor networks deployment techniques", *IEEE Communications Surveys & Tutorials*, vol. 16, no.2, pp. 834-854, 2014.

[22] K. Yetgin, T. K. Cheung, M. El-Hajjar and L. Hanzo, "A survey of network lifetime maximization techniques," *IEEE Communications Surveys & Tutorials*, early access article, 2017.

[23] M. Hefeeda and H. Ahmadi, "A probabilistic coverage protocol for wireless sensor networks," *Proc. of IEEE International Conference on Network Protocols (ICNP'07)*, pp. 41-50, 2007.

[24] B. Carter and R. Ragade, "An extensible model for the deployment of non-isotropic sensors," *Proc. of IEEE Sensors Applications Symposium (SAS'08)*, pp. 22-25, Feb. 2008.

[25] N. Aitsaadi, N. Achir, K. Boussetta and B. Gavish, "A gradient approach for differentiated wireless sensor network deployment," *Proc. of IFIP Wireless Days (WD'08)*, 2008.

[26] D. N. Ahmed, S.S. Kanhere and S. Jha, "Probabilistic coverage in wireless sensor networks," *Proc. of IEEE Conference on Local Computer Networks (LCN'05)*, pp. 681-688, 2005.

[27] A. Alfes, "Occupancy Grids: A stochastic spatial representation for active robot perception", *in Proc. of Conference on Uncertainty in Artificial Intelligence*, pp. 60-70, 1990.

[28] A. Alfes, "Using occupancy grids for mobile robot perception and navigation", *IEEE Computer Magazine*, vol. 22, no.6, pp. 46-57, 1989.

[29] S. S. Dhillon and K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks," *Proc. of IEEE Wireless Communications and Networking Conference (WCNC'03)*, pp. 1609-1614, 2003.

[30] L. Yuan, W. Chen and Y. Xi, "A review of control and localization for mobile sensor networks," *Proc. of World Congress on Intelligent Control and Automation, (WCICA'06)*, vol.2, pp.9164-9168, 2006.

[31] http://www-robotics.usc.edu/~robomote/

[32] Z. Ming, D. Xiaojiang and K. Nygard, "Improving coverage performance in sensor networks by using mobile sensors," *Proc. of IEEE Military Communications Conference (MILCOM'05)*, vol. 5, pp. 3335-3341, 2005.

[33] F. Li, S. Xiong and L. Wang, "Recovering coverage holes by using mobile sensors in wireless SENSOR networks," *Proc. of International Conference on Computational Intelligence and Security (CIS)*, pp.746-749, 2011.

[34] H. Pishro-Nik, K.S. Chan and F. Fekri, "On connectivity properties of large-scale sensor networks," *Proc. of 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON)*, pp. 498- 507, 2004.

[35] M. Cardei and J. Wu, "Coverage in wireless sensor networks", *in Handbook of Sensor Networks*, CRC Press, 2005.

[36] H. Zhang and J. Hou, "Maintaining sensing coverage and connectivity in large sensor networks", *Ad Hoc & Sensor Wireless Networks*, vol. 1, pp. 89-124, 2005.

[37] Y. Kim, C. Kim, D. Yang, Y. Oh and Y. Han, "Regular sensor deployment patterns for p-coverage and q-connectivity in wireless sensor networks," *Proc. of International Conference on Information Networking (ICOIN)*, pp.290-295, 2012.

[38] G. Xing, X. Wang, Y Zhang, C. Lu, R. Pless and C. Gill, "Integrated coverage and connectivity configuration for energy conservation in sensor networks," *ACM Transactions on Sensor Networks*, vol. 1, no. 2, pp.36-72, 2005.

[39] Y. Sun, Z. Yu, J. Ge, B. Lin and Z. Yun, "On deploying wireless sensors to achieve both coverage and connectivity," *Proc. of 5th International Conference on Wireless Communications, Networking and Mobile Computing (WiCom '09)*, 2009.

[40] X. Shen and J. Chen, Y. Sun, "Grid Scan: A simple and effective approach for coverage issue in wireless sensor networks," *Proc. of IEEE International Conference on Communications (ICC '06)*, pp.3480-3484, 2006.

[41] K. Chakrabarty, S.S. Iyengar, Q. Hairong and C. Eungchun, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Transactions on Computers*, vol. 51, no.12, pp. 1448-1453, 2002.

[42] J. H. Holland, *Adaption in Natural and Artificial Systems*, University of Michigan Press, 1975.

[43] R. Poli, W.B. Langdon, N.F. McPhee, *A Field Guide to Genetic Programming*, http://gp-field-guide.org.uk, 2008.

[44] M. Gen, R. Cheng, *Genetic Algorithms and Engineering Optimization*, John Wiley and Sons, 2000, pp. 2- 40.

[45] C. M. Fonseca, P. J. Fleming, "Genetic Algorithms for multi objective optimization: formulation, discussion and generalization", *Proc. of the International Conference on Genetic Algorithms*, pp 416-423, 1993.

[46] M. de Berg, O. Cheong, M. van Kreveld and M. Overmars, *Computational Geometry: Algorithms and Applications*, Springer, 3rd Edition, 2008.

[47] F. Aurenhammer, "Voronoi Diagrams – a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol.23, no.3, Sep. 1991.

[48] G. Wang, G. Cao and T. L. Porta, "Movement-assisted sensor deployment," *IEEE Transactions on Computers*, vol. 5, no.6, pp. 640-652, 2006.

[49] C. Kim, Y. Kim, Y. Han, H. Lee and Y. Jeong, "An energy-efficient self-deployment scheme in intelligent mobile sensor networks," *Proc. of International Conference on Multimedia and Ubiquitous Engineering (MUE)*, 2010.

[50] H. Lee, Y. Kim, Y. Han and C. Park, "Centroid-based movement assisted sensor deployment schemes in wireless sensor networks," *Proc. of IEEE Vehicular Technology Conference-Fall (VTC 2009-Fall)*, 2009.

[51] M. R. Ingle and N. Bawane, "An energy efficient deployment of nodes in wireless sensor network using Voronoi diagram," *Proc. of 3rd International Conference on Electronics Computer Technology (ICECT)*, vol.6, pp.307-311, 2011.

[52] A. Boukerche and X. Fei, "A voronoi approach for coverage protocols in wireless sensor networks," *Proc. of IEEE Global Telecommunications Conference (GLOBECOM '07)*, pp.5190-5194, 2007.

[53] S. Meguerdichian, F. Koushanfar, M. Potkonjak and M. Srivastava "Coverage problems in wireless ad-hoc sensor network," *Proc. of the Annual IEEE International Conference on Computer Communications (INFOCOM'01)*, pp. 1380-1387, 2001.

[54] X. Li, P. Wan and O. Frieder, "Coverage in wireless ad hoc sensor networks," *IEEE Transactions on Computers*, vol. 52, no .6, pp. 753-763, 2003.

[55] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots", *International Journal of Robotics Research*, vol. 5, pp. 90-98, 1986.

[56] R. Arkin, "Motor schema based navigation for a mobile robot: An approach to programming by behavior," *Proc. of IEEE International Conference on Robotics and Automation*, pp. 264-271, 1987.

[57] F.E. Schneider, D. Wildermuth, "A potential field based approach to multi robot formation navigation," *Proc. of IEEE International Conference on Intelligent Systems and Signal Processing*, pp. 680-685, 2003.

[58] W. H. Fan, Y. H. Liu, F. Wang, X.P. Cai, "Multi-robot formation control using potential field for mobile ad-hoc networks," *Proc. of IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp.133-138, 2005.

[59] M. Zhang, Y. Shen, Q. Wang, Y. Wang, "Dynamic artificial potential field based multi-robot formation control," *Proc. of IEEE Conference on Instrumentation and Measurement Technology*, pp.1530-1534, 2010.

[60] A. Howard, M. J. Mataric, G. S. Sukhatme, "Mobile sensor network deployment using potential field: a distributed scalable solution to the area coverage problem", *Proc. of International Symposium on Distributed Autonomous Robotics Systems (DARS'02)*, 2002.

[61] K.E. Parsopoulos, M.N. Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications*, IGI Global, 2010.

[62] F. Neumann, C. Witt, *Bio-inspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*, Springer, pp. 28-30, 2010.

[63] G. Beni, J. Wang, "Swarm intelligence in cellular robotic systems" In P. Dario, G. Sandini, P. Aebischer (Eds.), *Robotics and Biological Systems: Towards a New Bonics, NATO ASI Series, Series F: Computer and System Science*, pp. 703-712.

[64] R. Eberhart, J. Kermedy, "A New Optimizer Using Particles Swarm Theory", *Proc. of International Symposium on Micro Machine and Human Science*, pp. 39-43, 1995.

[65] Y. Shi, R. Eberhart, "A modified particle swarm optimizer," *Proc. of IEEE World Congress on Computational Intelligence*, pp.69-73, May 1998.

[66] E. Ozcan, C.K. Mohan, "Particle swarm optimization: surfing the waves," *Proc. of 1999 Congress on Evolutionary Computation*, pp. 1939-1944, 1999.

[67] H. Xiaohui, R. Eberhart, S. Yuhui, "Particle swarm with extended memory for multi-objective optimization," *Proc. of IEEE Symposium on Swarm Intelligence (SIS '03)*, pp. 193-197, 2003.

[68] M. Dorigo and T. Stutzle, *Ant colony optimization*, MIT Press, 2004, pp. 25-100.

[69] D.B. Jourdan, O.L. de Weck, "Layout optimization for a wireless sensor network using a multi objective genetic algorithm", *Proc. of IEEE Semi-annual Vehicular Technology Conference-Spring (VTC'04-spring)*, vol. 5, pp. 2466-2470, 2004.

[70] D.B. Jourdan, and O.L. de Weck, "Multi-objective genetic algorithm for the automated planning of a wireless sensor network to monitor a critical facility," *Proc. of SPIE Defense and Security Symposium*, pp. 565-575, 2004.

[71] I. Harvey, "The microbial genetic algorithm", *Evolutionary Computation*, 1996.

[72] B. Carter, R. Ragade, "A probabilistic model for the deployment of sensors", *IEEE Sensors Applications Symposium (SAS'09)*, pp. 7-12, 2009.

[73] Yong Xu and Xin Yao, "A GA approach to the optimal placement of sensors in wireless sensor networks with obstacles and preferences," *Proc. Of the IEEE Consumer Communications and Networking Conference (CCNC 2006)*, pp. 127-131, 2006.

[74] S.-S. Choi and B.-R. Moon, "Normalization for genetic algorithms with non-synonymously redundant encodings," *IEEE Transaction on Evolutionary Computing*, vol. 12, no. 5, pp. 604-616, 2008.

[75] D. S. Deif and Y. Gadallah, "Wireless sensor network deployment using a variable-length genetic algorithm", *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC 2014)*, pp. 2450-2455, 2014.

[76] O. Zorlu and O. K. Sahingoz, "Increasing the coverage of homogeneous wireless sensor network by genetic algorithm based deployment," *Proc. of International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, pp. 109-114, 2016.

[77] G. Wang, G. Cao, T. LaPorta, "A bidding protocol for deploying mobile sensors," *Proc. of IEEE International Conference on Network Protocol*, pp. 315-324, Nov. 2003.

[78] Z. Zhou, S. Wang, "Centroid optimization coverage and polling the sets of the coverage nodes: An improved coverage scheme," *Proc. of International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM'06)*, 2006.

[79] N. Heo, P.K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 1, no. 35, pp.78-92, 2005.

[80] C. H. Wu, K. C. Lee and Y. C. Chung, "A delaunay triangulation based method for wireless sensor network deployment," *Proc. of International Conference on Parallel and Distributed Systems (ICPADS' 06)*, 2006.

[81] D.O. Popa, H.E. Stephanou, C. Helm and A. C. Sanderson, "Robotic deployment of sensor networks using potential fields," *Proc. of IEEE International Conference on Robotics and Automation (ICRA '04)*, pp. 642-647, 2004.

[82] M. Senouci, M. Abdelhamid, K. Assnoune, "Localized movement-assisted sensor deployment algorithm for hole detection and healing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 5, pp. 1267-1277, 2014.

[83] M. Rout and R. Roy, "Self-deployment of mobile sensors to achieve target coverage in the presence of obstacles," *IEEE Sensors Journal*, vol. 16, no. 14, pp. 5837-5842, 2016.

[84] W. Jia, "Coverage enhanced algorithm using artificial potential force," *Proc. of International Conference on Intelligent Control and Information Processing (ICICIP'11)*, vol. 2, pp. 969-972, 2011.

[85] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," *Proc. of the Annual IEEE International Conference on Computer Communications (INFOCOM'03)*, vol. 2, pp.1293-1303, 2003.

[86] S. Li, C. Xu, W. Pan, Y. Pan, "Sensor deployment optimization for detecting maneuvering targets," *Proc. of 8th International Conference on Information Fusion*, vol.2, pp. 1629-1635, July 2005.

[87] X. Yu, W. Huang, J. Lan, X. Qian, "A novel virtual force approach for node deployment in wireless sensor network," *Proc. of IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 359-363, 2012.

[88] W. Xiaoling, S. Lei, Y. Jie, X. Hui, J. Cho, S. Lee, "Swarm based sensor deployment optimization in ad hoc sensor networks", *Proc. of International Conference on Embedded Software and Systems (ICESS'05)*, 2005.

[89] N. A. Aziz, A. W. Mohemmed, B. S. Daya Sagar, "Particle Swarm Optimization and Voronoi diagram for Wireless Sensor Networks coverage optimization," *Proc. of International Conference on Intelligent and Advanced Systems (ICIAS 2007)*, pp. 961-965, Nov. 2007.

[90] N. A. Aziz, A. W. Mohemmed, M. Y. Alias, "A wireless sensor network coverage optimization algorithm based on particle swarm optimization and Voronoi diagram," *Proc. of International Conference on Networking, Sensing and Control (ICNSC '09)*, pp.602-607, 2009.

[91] N. A. Aziz, A. W. Mohemmed, M. Y. Alias, K. A. Aziz, S. Syahali, "Coverage maximization and energy conservation for mobile wireless sensor networks: A two phase particle swarm optimization algorithm," *Proc. of International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2011)*, pp.64-69, 2011.

[92] H. Safa, W. El-Hajj, H. Zoubian, "Particle swarm optimization based approach to solve the multiple sink placement problem in WSNs," *IEEE International Conference on Communications (ICC)*, 2012, pp.5445-5450, 2012.

[93] D. Li, W. Liu and L. Cui, "EasiDesign: an improved ant colony algorithm for sensor deployment in real sensor network system," *IEEE Global Telecommunications Conference (GLOBECOM 2010)*, 2010.

[94] X. Liu, "Sensor deployment of wireless sensor networks based on ant colony optimization with three Classes of ant transitions," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1604-1607, 2012.

[95] G. Huang, D. Chen and X. Liu, "A node deployment strategy for blindness avoiding in wireless sensor networks," *IEEE Communications Letters*, vol. 19, no. 6, pp. 1005-1008, 2015.

[96] C. Zhu, C. Zheng, L. Shu and G. Han, "A survey on coverage and connectivity issues in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 619-632, 2012.

[97] J. Virkki, Y. Zhu, Y. Meng, and L. Chen, "Reliability of WSN hardware," *International Journal of Embedded Systems*, vol. 1, no. 2, 2012.

[98] http://www.tadiranbatteries.de/eng/products/lithium-thionyl-chloride-batteries/overview.asp

[99] https://www.omnisense.com/oms_cds/media/008-002-002%20OmniSense%20FMS%20Sensor%20Battery%20Life.pdf

[100] N. Baccour, A. Koubaa, L. Mottola, M. A. Zuniga, H. Youssef, C. A. Boano, and . Alves, “Radio link quality estimation in wireless sensor networks: a survey” *ACM Transactions on Sensor Networks*, vol. 8, no. 4, 2012.
[101] Y. Ali and S. Narasimhan “Sensor network design for maximizing reliability of linear processes,” *Journal of AIChE*, vol. 39, pp. 820–828, 1993.
[102] M. Bagajewicz and M. Sanchez, “Cost-optimal design of reliable sensor networks,” *El Sevier Journal of Computer and Chemical Engineering*, vol. 23, pp. 1757-1762, 2000.
[103] P. R. Kotecha, M. Bhushan, R. D. Gudi, and M. K. Keshari. "A duality based framework for integrating reliability and precision for sensor network design," *El Sevier Journal of Process Control*, vol. 18, no. 2 pp. 189-201, 2008.
[104] H. M. AboElFotoh, S.S. Iyengar and K. Chakrabarty, "Computing reliability and message delay for cooperative wireless distributed sensor networks subject to random failures," *IEEE Transactions on Reliability*, vol. 54, no. 1, pp. 145-155, 2005.
[105] Y. Jin, H. Lin, Zhang, Z. Zhang and X. Zhang, "Estimating the reliability and lifetime of wireless sensor network," *Proc. of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, 2008.
[106] C. Jaggle, J. Neidig, T. Grosch and F. Dressler, "Introduction to model-based reliability evaluation of wireless sensor networks," *Proc. of the International Federation of Automatic Control (IFAC) Workshop on Dependable Control of Discrete Systems*, pp. 149-154, 2009.
[107] E. I. Gokce, A. K. Shrivastava and Y. Ding. "Fault tolerance analysis of surveillance sensor systems," *IEEE Transactions on Reliability*, vol. 62, no. 2, pp. 478-489, 2013.
[108] A. Damaso, N.Rosa, P. Maciel, “Reliability of wireless sensor networks”, *Sensors*, vol.14, no.9, pp. 15760-15785, 2014.
[109] I. Silva, L. A. Guedes, P. Portugal and F. Vasques, "Reliability and availability evaluation of wireless sensor networks for industrial applications," *Sensors*, vol. 12, no. 1, pp. 806-838, 2012.
[110] I. Silva, L. A. Guedes, P. Portugal and F. Vasques, “A dependability evaluation tool for the Internet of Things”, *Computers & Electrical Engineering*, vol. 39, no.7, pp. 806-838, 2013
[111] http://www.reliabilityanalytics.com/blog/2011/09/02/reliability-modeling-k-out-of-n-configutation/
[112] M. O. Ball, "Computational complexity of network reliability analysis: An overview," *IEEE Transactions on Reliability*, vol. 35, no. 3, pp. 230-239, 1986.
[113] H. Van-Trinh, N. Julien and P. Berruet, “On-line self-diagnosis based on power measurement for a wireless sensor node” *Proc. of the IEEE Workshop on Highly-Reliable Power-Efficient Embedded Designs*, 2013.
[114] Y. Shpungin, "Combinatorial approach to reliability evaluation of network with unreliable nodes and unreliable edges," *International Journal of Computer Science*, vol. 1, no. 3, pp. 177-183, 2006.
[115] I. B. Gertsbakh and Y. Shpungin, *Models of Network Reliability: Analysis, Combinatorics, and Monte Carlo*, CRC Press, 1st edition, 2010, pp. 17-23.
[116] http://www.ti.com/product/CC2420/quality
[117] X. Chen, Y.S. Ong, M.H. Lim, K.C. Tan, “ A multi-facet survey on memetic computation” *IEEE Transactions on Evolutionary Computation*, vol. 15, pp. 591-607, 2011.
[118] P. Moscato and C.A. Cotta, “A modern introduction to memetic algorithms”, In *Handbook of Metaheuristics*, International series in operations research and management science, Gendreau, M., Potvin, J., Eds.; Springer, 2nd ed., 2010, pp. 141-183.
[119] J. Levine and F. Ducatelle, “Ant colony optimization and local search for bin packing and cutting stock problems,” *Journal of the Operational Research Society*, vol. 55, pp. 705-716, 2004.
[120] F. Neumann, D. Sudholt and C. Witt, “Rigorous analyses for the combination of ant colony optimization and local search,” *Proc. of International Conference on Ant Colony Optimization and Swarm Intelligence*, pp. 132-143, 2008.
[121] Y. Lin, J. Zhang, H. S. H. Chung, W. H. Ip, Y. Li and Y. H. Shi, "An ant colony optimization approach for maximizing the lifetime of heterogeneous wireless sensor networks," *IEEE*

Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 3, pp. 408-420, 2012.
[122] W. Ke, B. H. Liu and M.J. Tsai, "Constructing a WSN to fully cover critical grids by deploying minimum sensors on grid points is NP-complete", *IEEE Transactions on Computers*, vol. 56, no. 5, 2007.
[123] J.J. Schneider, S. Kirkpatrick, *Stochastic Optimization*, Springer, 2006, pp. 43-169.
[124] R.Wiener, "Branch and Bound implementations of the travelling salesman problem: Part1", *Journal of Object Technology*, vol.2, no.2, March 2003.
[125] N. B. Sariff and Norlida Buniyamin, "Comparative study of genetic algorithm and ant colony optimization algorithm performances for robot path planning in global static environments of different complexities," *Proc. of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pp. 132-137, 2009.
[126] E. Elbeltagi, T. Hegazy and D. Grierson. "Comparison among five evolutionary-based optimization algorithms," *El Sevier Journal of Advanced engineering informatics*, vol.19, no. 1 pp. 43-53, 2005.
[127] R. Putha, L. Quadrifoglio and Emily Zechman, "Comparing ant colony optimization and genetic algorithm approaches for solving traffic signal coordination under oversaturation conditions, " *Journal of Computer Aided Civil and Infrastructure Engineering*, vol.27, no. 1 pp. 14-28, 2012.
[128] V. Saishanmuga and S. P. Rajagopalan, "A comparative analysis of optimization techniques for artificial neural networks in bio-medical applications," *Journal of Computer Science*, vol.10, no. 1, pp. 106-114, 2014.
[129] K. M. Rana and M. A. Zaveri, "Techniques for efficient routing in wireless sensor network," *Proc. of International conference on Intelligent Systems and Data Processing (ICISD)*, 2011.
[130] R. Haupt and S. Haupt, *Practical Genetic Algorithms*, Wiley, 2nd edition, 2004, pp. 27- 64.
[131] C. C. Lai, C. K. Ting, R. S. Ko, "An effective genetic algorithm to improve wireless sensor network lifetime for large-scale surveillance applications," *Proc. of the IEEE Congress on Evolutionary Computation (CEC'07)*, pp. 3531–3538, 2007.
[132] A. Sinha, Y. Chen, D. E. Goldberg, "Designing efficient genetic and evolutionary algorithm hybrids," in *Studies in Fuzziness and Soft Computing*, Springer, 2004, pp. 259-288.
[133] A.E. Eiben and K. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol.1, no.1, pp.19-31, 2011.
[134] R. Haupt and S. Haupt, *Practical Genetic Algorithms*, Wiley, 2nd edition, 2004, pp. 128-135.
[135] Wen Wan and Jeffrey B. Birch, "An improved hybrid genetic algorithm with a new local search procedure," *Journal of Applied Mathematics*, vol. 2013, Article ID 103591, 10 pages, 2013. doi:10.1155/2013/103591.
[136] E. A. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, vol. 53, Heidelberg: Springer, 2003.
[137] D.C. Montogomery, *Deisgn and Analysis of Experiments*, Wiley, 8th edition,2013,pp. 26-52.
[138] M. Li, Z. Li and A. V. Vasilakos, "A survey on topology control in wireless sensor networks: taxonomy, comparative study, and open Issues," *Proceedings of the IEEE*, vol. 101, no. 12, pp. 2538-2557, 2013.
[139] F. Delicato, F. Protti, L. Pirmez and J.F. de Rezende,"An efficient heuristic for selecting active nodes in wireless sensor networks", *Computer Networks*, vol. 50, no. 18, pp. 3701-3720, 2006.
[140] A. Nagpur and S. Patil, "Topology control in wireless sensor networks: An overview," *International Journal of Computer Applications*, vol. 92, no. 7, 2014.
[141] Yi Zou and K. Chakrabarty, "A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 978-991, 2005.
[142] D. Tian and N.D. Georganas, "A node scheduling scheme for energy conservation in large wireless sensor networks," *Wireless Comm. and Mobile Computing*, vol. 3, pp. 271-290, 2003.
[143] Y. Li, Yingshu, C. Ai, Z. Cai and R. Beyah, "Sensor scheduling for p-percent coverage in wireless sensor networks," *Cluster Computing*, vol. 14, no. 1, pp. 27-40, 2011.

[144] H. P. Gupta, S. V. Rao and T. Venkatesh, "Sleep scheduling for partial coverage in heterogeneous wireless sensor networks," *Proc. of the International Conference on Communication Systems and Networks (COMSNETS)*, 2013.
[145] M. Hefeeda and H. Ahmadi, "Energy-efficient protocol for deterministic and probabilistic coverage in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 5, pp. 579-593, 2010.
[146] Y. Xu, J. Heidemann and D. Estrin, "Geography-informed energy conservation for ad hoc routing", *Proc. of International Conference on Mobile Computing and Networking (MOBICOM)*, 2001.
[147] F. Ye, G. Zhong, S. Lu and L. Zhang, "PEAS: A robust energy conserving protocol for long-lived sensor networks", *Proc. of the International Conference on Distributed Computing Systems (ICDCS)*, 2003.
[148] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "SPAN: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Proc. Of MobiCom*, pp. 85-96, 2001.
[149] S. Gao, C. Vu and Y. Li, "Sensor scheduling for k-coverage in wireless sensor networks," *Proc. Of the International Conference on Mobile Ad-hoc and Sensor Networks*, 2006.
[150] M. Cardei and D. Z. Du, "Improving wireless sensor network lifetime through power aware organization," *Wireless Networks*, vol. 11, no. 3, pp. 333-340, 2005.
[151] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," *in Proc. of the IEEE International Conference on Communications (ICC)*, 2001.
[152] C. C. Lai, C. K. Ting and R. S. Ko, "An effective genetic algorithm to improve wireless sensor network lifetime for large-scale surveillance applications," *Proc. of the IEEE Congress on Evolutionary Computation*, pp. 3531–3538, 2007.
[153] https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja.
[154] http://anrg.usc.edu/contiki/index.php/Cooja_Simulator
[155] A. Dunkels, B. Gron¨ vall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," *Proc. of the First IEEE Workshop on Embedded Networked Sensors*, 2004
[156] http://www.contiki-os.org/
[157] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA" *Proc. of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, 2006.
[158] http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf
[159] https://tools.ietf.org/html/rfc6550
[160] S. Dasa, S.S. Mullicka and P.N. Suganthanb, "Recent advances in differential evolution – An updated survey," *Elsevier Journal on Swarm and Evolutionary Computation*, vol. 27, no. 4, 2016.
[161] M. Nikolić and D. Teodorović, "Empirical study of the Bee Colony Optimization (BCO) algorithm", *Elsevier Journal on Expert Systems with Applications*, vol. 40, no. 11, , pp. 4609-4620, 2013.
[162] https://www.ietf.org/rfc/rfc3561.txt