University of Sheffield

Optimising IDS Configurations for IoT Networks using AI Approaches

Abdulmonem Alshahrani

Supervisors:

Prof. John A. Clark
Prof. Hamish Cunningham

A thesis submitted for the degree of *Doctor of Philosophy*

in the

Department of Computer Science
The University of Sheffield

July 24, 2023

Declaration

Abstract

The number of internet-connected smart objects, known as the Internet of Things (IoT), has increased significantly in recent years. The low cost of manufacturing has enabled a proliferation of smart devices across many tasks and domains. Such devices, however, are typically resource constrained. This has led to the emergence of Low-Power and Lossy Networks (LLNs) which require efficient communication protocols. The Routing Protocol for Low-Power and Lossy Networks (RPL) has been designed for such a purpose. The RPL is the *de-facto* standard routing protocol for the IoT. Nevertheless, RPL-enabled networks are susceptible to many attacks as these devices are unattended, resource-constrained, and connected via unreliable networks.

Deploying Intrusion Detection Systems (IDSs) in such a large and resource-constrained environment is a challenging task. The resource-constrained nature of many devices and nodes restricts what tasks those nodes can realistically expect to perform. There may be a great many choices as to what detection functionality is allocated and where. There are cost/benefit trade-offs between them and inappropriately favouring one over the another may cause an ineffective IDS deployment. In this research, we investigate the use of a meta-heuristic-based optimisation method, namely a Genetic Algorithm (GA), to discover optimal IDS placements and configurations for the Low Power and Lossy Networks (LLNs). To the best of our knowledge, this is the first attempt to optimise IDS configurations for emerging and constrained networks while incorporating a wider set of aspects than currently considered. Our approach seeks to optimise and balance detection performance (either detection rate or $F1$ score), coverage (nodes are monitored by an appropriate number of probes), feasibility cost (nodes host detection functionality within their capability), and deployment cost (seeking to reduce the number of probes deployed). We propose a framework that makes trades-offs between these functional and non-functional constraints.

A genetic algorithm-based optimisation approach is developed to address the IDS optimisation task. However, the fitness function is evaluated in part via a computationally expensive simulation. We show how a neural network can be used as a surrogate fitness function evaluation, providing better results more cheaply. Experimental results show that the proposed function approximation is more computationally efficient. Our approximation-based GA system is 1.6 times faster than the corresponding simulation-based GA system. It also gives better results. Furthermore, when used repeatedly to generate candidate placements and configurations the resource costs per generation reduce drastically.

The surrogate model is valuable as it significantly reduces the evaluation time and computation. However, generality is still a limitation. Therefore, we propose a transfer-learning Deep Neural Networks (DNNs) approach, that harnesses the experience of previously trained neural networks, to develop a general proxy model for evaluating IDS configurations of variant newly-presented networks more accurately.

Acknowledgement

First and foremost, I thank God (Allah) who has given me strength throughout all the challenging moments of completing this Ph.D. study.

I would like to express my deepest gratitude to my supervisor, Professor John A. Clark, who generously provided me with knowledge and expertise, and guided me throughout this research. Further, I extend my appreciation to my panel members, Professor Hamish Cunningham (my second supervisor) and Dr Prosanta Gope (panel chair), for their valuable feedback during my panel meetings.

I would be neglectful if I don't mention my parents, my wife, my son, my daughter, my brothers and sisters, and my friends. Their existence, prayers and belief in me have kept my spirits and motivation high during this entire journey.

I would like to extend my sincere thanks to my labmates in the Security of Advanced Systems research group for being amazing people to work with.

Lastly, I am also grateful to my scholarship provider, King Khalid University. This endeavour would not have been possible without their support.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

*R*2	**R**-**S**quare
6BR	IPv**6** **B**oarder **R**outer
6LoWPAN	IPv**6** over **Lo**w power and **W**ireless **P**ersonal **A**rea **N**etworks
ACK	**ACK**nowledgement
AES	**A**dvance **E**ncryption **S**tandard
AI	**A**rtificial **I**ntelligence
ANNs	**A**rtificial **N**eural **N**etworks
AODV	**A**d hoc **O**n-Demand **D**istance **V**ector Routing
AUC	**A**rea **U**nder the **C**urve
CCI	**C**orrectly **C**lassified **I**nstances
CEP	**C**omplex **E**vent **P**rocessing
CIDS	**C**entralised **I**ntrusion **D**etection **S**ystem
CoAP	**Co**nstrained **A**pplication **P**rotocol
DAG	**D**irected **A**cyclic **G**raph
DAO	**D**estination **A**dvertisement **O**bject
DDS	**D**ata **D**istribution **S**ervice
DIDS	**D**istributed **I**ntrusion **D**etection **S**ystem
DIO	DODAG **I**nformation **O**bject
DIS	DODAG **I**nformation **S**olicitation
DNNs	**D**eep **N**eural **N**etworks
DODAG	**D**estination **O**riented **D**irected **A**cyclic **G**raph
DoS	**D**enial **o**f **S**ervice
DPO	**D**istress **P**ropagation **O**bject
DS	**D**etection **S**niffers
DTR	DIS **T**ransmitting **R**ate
DT	**D**ecision **T**ree
EA	**E**volutionary **A**lgorithm
EC	**E**volutionary **C**omputations
EMS	**E**vent **M**anagement **S**ystem
EOP	**E**xpensive **O**ptimisation **P**roblem
ETX	**E**xpected **T**ransmission **C**ount
FNN	**F**eedforward **N**eural **N**etwork
FNR	**F**alse **N**egative **R**ate
FN	**F**alse **N**egative
FPR	**F**alse **P**ositive **R**ate
FP	**F**alse **P**ositive
GA	**G**enetic **A**lgorithm
GPS	**G**lobal **P**ositioning **S**ystem
GP	**G**enetic **P**rogramming
HIDS	**H**ost-based **I**ntrusion **D**etection **S**ystems
HPC	**H**igh **P**erformance **C**omputing
HR	**H**igh **R**esourced
IA	**I**nformation **A**ssurance
ICS	**I**ndustrial **C**ontrol **S**ystems
IDS	**I**ntrusion **D**etection **S**ystem
IETF	**I**nternet **E**ngineering **T**ask **F**orce
IoT	**I**nternet of **T**hings
IPsec	**I**nternet **S**ecurity **P**rotocol
IPS	**I**ntrusion **P**revention **S**ystem
LLN	**L**ow **P**ower and **L**ossy Network

LR	**L**ow **R**esourced
MAE	**M**ean **A**bsolute **E**rror
MAPE	**M**ean **A**bsolute **P**ercentage **E**rror
ML	**M**achine **L**earning
MOGA	**M**ulti **O**bjective **G**enetic **A**lgorithm
MOO	**M**ulti **O**bjective **O**ptimisation
MQTT	**M**essage **Q**ueue **T**elemetry **T**ransport
MR	**M**oderate **R**esourced
MSE	**M**ean **S**quare **E**rror
NFC	**N**ear **F**ield **C**ommunication
NIDS	**N**etwork-based **I**ntrusion **D**etection **S**ystems
NN	**N**eural **N**etwork
NS2	**N**etwork **S**imulator 2
NSGA	**N**on-Dominated **S**orting **G**enetic **A**lgorithm
OF	**O**bjective **F**unction
PDR	**P**acket **D**ropping **R**atio
RBAC	**R**ule **B**ased **A**ccess **C**ontrol
RBF	**R**adial **B**asis **F**unction
ReLU	**Re**ctified **L**inear **U**nit
RFID	**R**adio **F**requency **I**dentifier
RFR	**R**andom **F**orest **R**egression
RMSE	**R**oot **M**ean **S**quare **E**rror
RNG	**R**andom **N**etwork **G**enerator
RPL	**R**outing **P**rotocol for **L**ow Power and Lossy Network
RS	**R**andom **S**earch
SCADA	**S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition
SLA	**S**ervice **L**evel **A**greement
SMOGN	**S**ynthetic **M**inority **O**ver-sampling with **G**aussian **N**oise
SMOTER	**S**ynthetic **M**inority **O**ver-sampling **T**echnique for **R**egression
SMOTE	**S**ynthetic **M**inority **O**ver-sampling **T**echnique
SN	**S**uper **N**ode
SOO	**S**ingle **O**bjective **O**ptimisation
SPEA	**S**trength **P**areto **E**volutionary **A**lgorithm
SVM	**S**upport **V**ector **M**achine
SVR	**S**upport **V**ector **R**egression
TN	**T**rue **N**egatives
TP	**T**rue **P**ositives
VANET	**V**ehicular **A**d hoc **NET**work
WBGA	**W**eight **B**ased **G**enetic **A**lgorithm
WSN	**W**ireless **S**ensor **N**etworks

Symbols

$\hat{y}$	predicted value
B	number of bins
b	granularity value
C	Evaluation cost
cov	coverage value
D	number of samples
dBm	decibel-milliwatts
f	fitness value
fit	fitness function
$freq$	frequency
g	counter
k	target of monitoring nodes
l	number of participants for GA selection
m	number of neighbouring nodes
m_{rj}	maximum number of rules
mJ	megajoule
ms	milliseconds
N	Number of evaluation
n	number of nodes
np	number of packets
O	Time complexity
P	list containing configurations
p	probability value

PL	parallel computing	T	threshold value
R	number of rules	Thr	threshold bits
r_j	current number of rules	v	number of monitoring nodes
S	set of nodes	W	wight value
s	seconds	y	actual value

Dedicated to my family, without whose endless love and support, I could not achieve this.

Chapter 1

Introduction

Computer and network security is an ever-growing research area. There are many proposals and techniques to enhance protection against malicious activities, for example, access control, authentication, firewalls and others. However, an attacker on occasion may bypass these mechanisms and, hence, there is a need for a second line of defence.

The first attempt to detect intrusions was in 1980 by Anderson [6]. He suggested methods to analyse audit logs for signs of intrusion. This sort of system would become known as an Intrusion Detection System (IDS). Heberlein et al.[7] extended the idea to scrutinise not only historic logged data in a host-based manner but also stream data to monitor the network traffic for any security violations.

Even though intrusion detection systems have succeeded in detecting various threats in an efficient and effective way in traditional networks, the emergence of new paradigms such as ubiquitous computing, embedded systems and wireless communications have introduced further difficulties [8].

The Internet of Things (IoT), as one type of rapidly growing and adopted modern network, includes a large number of interconnected and heterogeneous devices with limited resources in terms of power, computation, storage and communication capabilities [9, 10]. Such networks are known as low-power and lossy networks (LLNs). Resource efficiency is a critical issue in some systems. Routing efficiency has emerged as a particularly important problem for

IoT LLNs. The Routing Protocol for Low-power and lossy networks (RPL) is proposed as the IETF standard IPv6 routing protocol for large-scale LLNs [11]. However, the devices in LNNs are resource-constrained. This means these devices may not be able to support complex defence mechanisms. RPL nodes are not tamper-resistant and are vulnerable to several attacks[1].

These limitations in capabilities have to be taken into consideration when designing an IDS for RPL-based IoT because they may affect its detection capabilities. One of the least studied fields in IDS is how to determine optimal configurations where not all the nodes are sufficiently well-resourced to host sophisticated (or even any) IDS functionality [12]. In this PhD research, we investigate the use of a specific type of optimisation approach, namely Evolutionary Algorithms (EAs), to find effective and efficient IDS sensors[2] placement and configuration for the RPL-based IoT constrained networks. We also investigate means of decreasing the resources needed to perform optimisation searches.

1.1 Problem statement and motivation summary

IoT is emerging as one of the most exciting architectural developments in modern-day networking technology [13]. The provision of intrusion detection services for RPL-based IoT is a natural development that researchers have begun to address. Many of the state-of-the-art IDS proposals for the RPL-based IoT tend to focus on the functional criteria such as increasing the detection rate and reducing false alarms. These are (unsurprisingly) prominent evaluation metrics for intrusion detection systems. However, in a constrained and vulnerable network, other non-functional metrics are important too. The non-functional metrics include the study of where these IDS sensors are placed, what are the resources available on the hosted nodes, and how many monitoring nodes are needed to maintain resiliency.

There is a real issue as to how high-performing IoT/RPL IDS deployments may be determined. One means is via optimisation-based approaches. Optimisation methods have been

[1]The terms attack, adversary, intruder, and malicious node are used interchangeably throughout this thesis
[2]The terms IDS sensor, probe, and monitoring node are used interchangeably throughout this thesis

widely applied in many real-world complex problem domains. They explore relevant search spaces seeking to optimise a fitness function or minimise a cost function. Most real-world applications involve complicated factors and parameters to determine how a system performs [14]. The search landscape will often be non-linear and require nonlinear optimisation approaches to be applied. These often require significant computing power [15]. This is indeed the case with application to IDSs. This is particularly so when the usefulness of a specific candidate IDS configuration is evaluated via a simulation. Attacks are launched on a simulation of the system to be protected and measures are taken as to how a candidate IDS performs. The attacks will generally follow some hypothesised distribution. A single evaluation may require the simulation of a significant number of attacks. Using a simulation environment in the context of a nonlinear search may be computation-expensive and time-consuming. Resource usage is important and we need to minimise the amount of effort involved in determining a high-performing configuration.

In this PhD thesis, we aim to provide an optimisation-based framework to produce high-performing IDS configurations for RPL-based IoT networks, and do so rapidly. To make this practical and much more useful, we are going to address the efficiency and effectiveness aspects.

1.2 Research hypotheses

IoT is a complex distributed network with resource-deficient electronic devices. The environment is known to be self-configured and mostly operates unattended [16]. RPL-based IoT networks are susceptible to both cyber (logical) and physical attacks. Finding optimal IDS configurations, in terms of both the functional (e.g., detection rate) and non-functional (e.g., deployment cost) criteria, for such a constrained and vulnerable network is an important and difficult task. We investigate whether evolutionary optimisation can automatically determine high-performing IDS configurations. Evolutionary approaches have proven to be an effective tool for deriving intrusion detection components [17]. However, there is no comprehensive application in the configuration of IDS for the RPL-based IoT ecosystem. The first hypothesis of this thesis is as follows:

Hypothesis 1: *Evolutionary algorithms can discover resource-efficient and detection-capable security configurations for intrusion detection systems that are suitable for RPL-based Internet of Things networks.*

Even though Evolutionary Algorithms (EAs) can provide us with optimal (or near-optimal) IDS configurations and be able to reconfigure them as needed, it is time-consuming and computation-expensive. EAs often require thousands of function evaluations to locate a near-optimal solution[3][18]. We propose that the computation complexity of function evaluation can be radically reduced by seeking to employ *function approximation*. One technique is by using Artificial Neural Networks (ANNs) to learn the underlying function via mapping inputs to outputs using historical or available observations from the domain. The ANNs can approximate a complex or unknown function with high-fidelity [19, 20]. Accordingly, we propose a Feedforward Neural Network (FNN) technique for IDS configuration function approximation. The second hypothesis is as follows:

Hypothesis 2: *Machine learning approaches can allow us to perform function approximation for the framework's fitness evaluation function and so greatly reduce the time and computation taken to produce near-optimal security configurations using such a framework.*

Accelerating the optimisation evaluation of the IDS configurations for a network is advantageous and important. However, when dynamically changing environments (or in general, newly presented networks) require optimising, the costly optimisation process needs to be started from scratch again. It is, therefore, beneficial to build an approximation model (a fast proxy) that can make accurate fitness evaluations not only for one specific network but also for new networks. As such, there is a need for a generalised function approximation technique. The third hypothesis is given next:

[3]The terms chromosomes, individuals, solutions and candidate solutions are used interchangeably throughout this thesis

Hypothesis 3: *A transfer learning based deep neural networks approach can provide a highly efficient fitness approximation with acceptable fidelity for newly-presented RPL-based Internet of Things networks.*

Thus, we seek to develop an approach whose fitness approximation is not tied to a specific network, i.e. it is more generally applicable.

1.3 Contributions

This thesis has the following contributions:

- *An IDS optimisation framework for RPL-based constrained networks.* This finds optimal placements and configurations of the IDS sensors. It incorporates a much wider set of constraints than other IoT IDS researchers consider. The studied objectives in this research are comprehensive and valuable for any IDS deployment. Further, the framework is extendable and further objectives can be included if desired.
- *A function approximation surrogate model.* This provides an extremely efficient approximate evaluation of IDS configurations for the network under consideration, greatly reducing the resources required for an IDS configuration evaluation.
- *A generalised proxy model.* We extend the capability of the function approximator model using the transfer-learning method to make possible fast, efficient and accurate IDS configurations evaluation for new variant networks.

1.4 Structure of the thesis

The rest of the thesis is organised as follows:

- In Chapter 2, we start with a brief background of the IoT and RPL. We then cover some security issues of these technologies. We also present a detailed description of the IDSs and related works concerning the deployment of the IDS in RPL-based IoT networks. Further, we address some optimisation issues of IDS configurations and the usage of evolutionary algorithms to solve the problem. Then we describe the function approximation concept and its usage in fitness evaluation.

- In Chapter 3, we summarise the originality of the research contributions given the literature review, indicating how the research hypotheses follow from it.

- In Chapter 4, we present a Genetic Algorithm (GA)-based IDS optimisation framework. This includes the architecture, implementation and evaluation of our optimisation framework. This chapter details the building blocks of the fitness function and objectives to be optimised the configuration of IDS sensors for RPL-based constrained networks.

- In Chapter 5, a surrogate-assistant-based Neural Network (NN) to accelerate the optimisation-based discovery of IDS configurations is described. We show how a Feed-forward Neural Networks (FNNs) model is built, trained and adopted to replace the expensive evaluator for the IDS configurations.

- In Chapter 6, we discuss the usage of a transfer-learning Deep Neural Network (DNN) to overcome the generalisation issue (i.e., a surrogate-model trained to make IDS evaluation for only one specific network). This is to generate high-performing IDS configurations for different networks. We show how a transfer-learning DNN model performs better than a stand-alone DNNs model (and other ML-based regression models) for enabling IDS configuration evaluation with reduced error.

- In Chapter 7, we evaluate the evidence produced in our investigation of the research hypothesis. We present a summary of the thesis and identify potential future work.

1.5 Publications

Research in this thesis has appeared in the following publications:

- *On Optimal Configuration of IDS for RPL Resource-Constrained Networks Using Evolutionary Algorithm.*

Alshahrani, A., Clark, J.A.
Future Technologies Conference (FTC) 2022, Volume 2. FTC 2022. Lecture Notes in Networks and Systems, vol 560. Springer, Cham. https://doi.org/10.1007/978-3-031-18458-1_35

- *Neural Network Approximation of Simulation-based IDS Fitness Evaluation*
  Alshahrani, A., Clark, J.A.
  25th IEEE International Conference on Computational Science and Engineering (IEEE CSE 2022).

- *Transfer Learning Approach to Discover IDS Configurations Using Deep Neural Networks*
  Alshahrani, A., & Clark, J. A. (2022, October). In 2022 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI) (pp. 1-8). IEEE.l Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI 2022).

The following paper is a collaboration work with others in the Security of Advanced Systems research group:

- *Intrusion detection systems in RPL-based 6LoWPAN: a systematic literature review*
  Pasikhani, A. M., Clark, J. A., Gope, P., & Alshahrani, A.
  IEEE Sensors Journal,2021.

Chapter 2

Literature Review

2.1 Overview of IoT

The world is much smarter thanks to the Internet of Things (IoT). It is defined as the collection of connected uniquely identifiable smart objects that communicate via the global internet. IoT services have increased rapidly in recent years and are set to continue to play a significant role [21]. The number of IoT-connected devices is expected to reach 83 billion by 2024 [22]. These smart objects and technologies have assisted companies increasing revenue and lowering costs, creating \$200−\$500 billion in profits per year by 2025 [23]. The IoT ecosystem connects people, machines, tablets, smartphones and other smart objects via very large-scale interconnected networks to innovate intelligent applications and services.

Some emerging applications are the smart home, smart city, smart health, cyber-physical systems, smart transportation, connected cars and smart grids. The *things* are physical sensors and actuators that gather information about the social life of humans or the environment in general to provide intelligent services. The IoT enables communication between heterogeneous smart devices, automobiles, fridges, ovens and other items or things through different networks at any time and from anywhere [24]. The flexible connectivity between humans and things is provided by the so-called 6A vision: Anywhere, Anytime, Anyone, Anything, Any network/path and Any service.

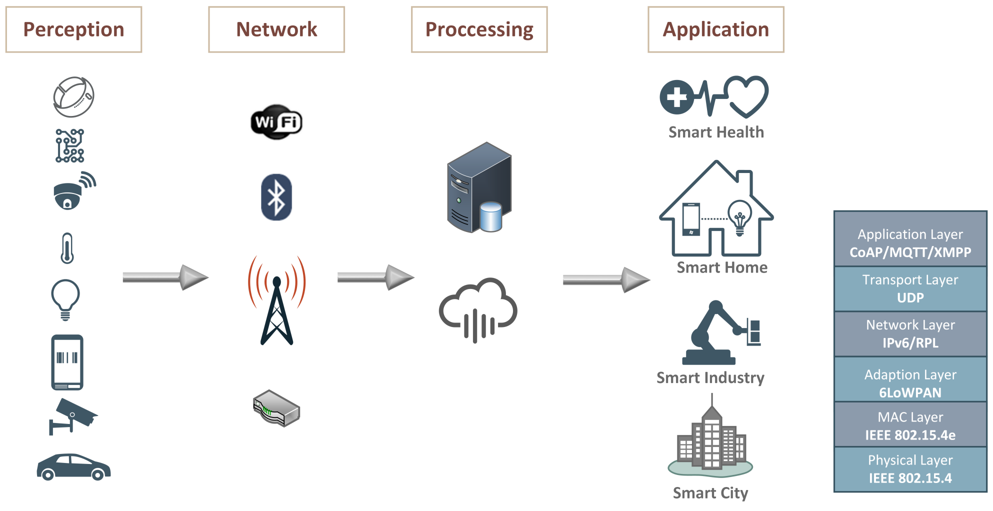


Figure 2.1: *Internet of Things architecture and protocol stacks*

Due to the rapid increase in the number of interconnected IoT smart objects and the resource-constrained nature of these nodes, The Internet Engineering Task Force (IETF) has formed a Working Group (WG) called ROLL (Routing Over Low power and Lossy networks) to work on the standardisation of efficient protocols to support IPv6 communication across wireless sensor or IoT networks. The standard routing protocol for the Internet of Things is the RPL (Routing Protocol for Low Power and Lossy Networks)[8]. It is an efficient routing protocol designed for Low Power and Lossy networks (LLNs). These networks include resourced-constrained nodes in terms of memory, power and CPU processing. In section 2.2, we will provide more details about the RPL protocol.

2.1.1 IoT architecture and layers

There are several proposed architectures of the IoT ecosystem; however, there is no agreed standard [25]. The most common architecture consists of four layers as shown in Figure 2.1. These are Perception Layer, Network Layer, Processing Layer and Application Layer. A brief introduction to each one is provided next.

2.1.1.1 Perception layer

This layer involves the physical devices and actuators that connect and communicate with each other. Most nodes or devices here are sensors, which is why this layer is sometimes called the sensing layer. They collect and gather information about the surrounding environment such as humidity, temperature and personal movements [26]. Identifying the analogue data and then digitising it is basically how this layer works. Some of the adopted technologies at this level are Radio Frequency Identifier (RFID) tags, Global Positioning System (GPS), sensor gateways, temperature sensors, and surveillance cameras. In some studies, this layer is further divided into two sub-layers: perception nodes (responsible for controlling data) and network nodes (used to send data to the controller and which connect the network layer)[27].

2.1.1.2 Network layer

This layer is above the sensors level and responsible for transmitting the acquired and digitised data from the sensors layer to the upper level. It is network communication software and physical components with receiving, transmitting and routing capabilities [9]. Data can be transmitted through a wired or wireless channel; some of these technologies are 3G, DSL, LTE, WiFi, Near-Field communication (NFC), and Bluetooth. Some of the routing protocols that have been adopted in this layer are the RPL and AODV (ad hoc on-demand distance vector).

2.1.1.3 Processing layer

Before information reaches end users, it has to be processed through this supporting layer. It is where a tremendous amount of data is processed, analysed and stored [28]. Some main technologies involved are cloud computing and big data processing [29]. Some researchers have combined it with the application layer since it works near the applications demanded by users.

2.1.1.4 Application Layer

This is where users and organisations visually interact and communicate with the gathered information via the services provided. Many IoT applications enable smart cities, smart transportation, smart healthcare, smart homes etc. The application layer supports standard transfer protocols such as CoAP (Constrained Application Protocol), MQTT (Message Queue Telemetry Transport), Extensible Messaging and Presence Protocol (XMPP) and DDS (Data Distribution Service). These protocols enable the constrained devices that run on low-power networks to communicate efficiently.

2.1.2 Securing the IoT

Integrating smart objects and devices with the physical world poses many security issues. Daily activities are sensed, recorded, processed, and transmitted through untrusted networks creating an excellent opportunity (i.e., multiple attack surfaces) for attackers to intercept information and destroy resources [29].

Attacks on critical systems such as Supervisory Control and Acquisition Systems (SCADA) and Industrial Control Systems (ICS), where IoT has been embraced, will have hazardous consequences for whole cities or even countries [30]. The IoT-embedded devices are shipped with weak configurations (e.g. with default passwords) and provide critical functions (e.g. door locking). Compromising these devices became easy and serious. For instance, tests on popular smart home appliances show the vulnerability of these devices to being compromised and leaking some sensitive personal information [31]. The adversary can exploit the wireless medium that connects these devices and exfiltrate information from the payload of the transmitted packets. This may reveal some information about the users and their activities.

2.1.2.1 Securing IoT versus conventional networks

There are several key security and privacy differences between IoT networks and other wired networks. IoT networks are deployed over Low Power and Lossy Networks (LLNs). In contrast, other networks employ more powerful devices and have secured communication

[32]. Embedded devices in LLNs are resource constrained. They are limited in processing capability, storage capacity and power supply (mostly battery-based) [13]. For traditional networks, these constraints are not particularly important considerations. In LLNs, however, inadequate computation power and insufficient storage capacity make applying cryptography solutions and other high-level security mechanisms impractical [32]. Lightweight encryption and security mechanisms are preferred to secure communication between IoT nodes and the global internet.

Additionally, the IoT era brings with it novel communication protocols such as the CoAP, which is similar in function to the HTTP in conventional networks, and the MQTT, which utilises a publish/subscribe mechanism that supports energy-efficient communication. Another unstable (in terms of security development) protocol is the RPL which directs the traffic flows efficiently between the tiny devices. IPv6 over Low power and Wireless Personal Area Networks (6LoWPAN) enables constrained devices to connect to the Internet through IPv6 standard [33]. These newly developed communication protocols and standards require further effort regarding security.

The most obvious building blocks of the IoT ecosystem are Wireless Sensor Networks (WSNs). WSNs are mostly used for gathering data such as monitoring and surveillance services. The collected data is transmitted to a sink node via multi-hop communication. Communication is mostly in one direction (the reverse direction serves principally to manage the sensor) and does not seek to have any effect in the physical world (i.e., largely reading information and passing it back) [29]. WSNs often consist of homogeneous sensors whereas heterogeneity is common in IoT systems [34]. This raises interconnection and security challenges. Furthermore, WSNs are peer-to-peer wireless networks where each WSN is mostly separated from other WSNs and enables different applications. Contrarily, the IoT connects many domains, applications, autonomous systems, and ad hoc networks, with WSNs as a part of it (often at a very large scale). In IoT networks, the sensors are globally identifiable by IPv6 addresses [8], which means they can interact with external agents. Table 2.1 outlines the main differences between IoT and WSN.

Attacks on IoT and WSN could be similar. However, the aforementioned extra features of the IoT network make designing effective security solutions a tricky task.

Table 2.1: *The difference in characteristics between IoT and WSN*

Characteristics	IoT	WSN
Physical world Coupling	High	Medium
Power constraints	Medium	High
Memory constraints	Medium	high
Heterogeneity	High	Low
Scalability	High	Medium
Mobility	High	Low
Privacy	High	Medium
Globally identifiable	High	Low

2.1.3 Security challenges in IoT

It is important to develop and improve IoT security solutions to protect sensitive data and critical information, and to gain user and industry confidence. However, there are some critical challenges as described in the following subsections.

2.1.3.1 Integration

The cyber-world is intertwined with the physical world. Any threats to any part of the IoT system will affect (have a negative impact on) the whole process. Another integration issue is when IoT applications are integrated with Cloud computing [35]. This coupling raises security concerns such as the trust of service providers, knowledge about the Service Level Agreement (SLA) and the physical location of data. In addition, there are some cases where legacy devices (with an old OS version) are still used and not updated [36]; that weak link could be used to compromise the cyber world. There is a need to study access control techniques that restrict the rapid propagation of security breaches [37].

2.1.3.2 Heterogeneity

Most IoT appliances run across a diverse range of operating systems deployed on different hardware specifications using various communication channels [38]. Each one has its own security methods and mechanisms, and such heterogeneity makes conventional security solutions unsuited to IoT systems. IoT networks consist of different devices with different computation capabilities. According to [39], a smartphone is capable of running an advanced learning method while a simple sensor can only perform a simple learning method. Consequently, the security level and learning outcomes required for each part of an IoT system differ. These multiple devices are provided by many companies with a variety of operating systems that require different security solutions.

Seamless interoperability is another related challenge. Interconnected devices are made of different entities connected through different communication protocols. They still need to function and interact as requested, no matter the circumstances. All levels and components of the IoT ecosystem have to be secured efficiently regardless of their hardware and software capability.

There are many potential entry points for attackers to break into and compromise an IoT-based system. Adversaries may choose to attack via the weakest link and so a comprehensive approach to security is needed. Securing IoT-based systems is often very hard.

2.1.3.3 Resource constraints

There will be more than 83 billion connected IoT devices on the Internet by 2024 [10]. The reason behind the rapid spread of IoT systems is that the companies that produce these devices make them low-cost yet resource-constrained. They have small memory space, low computation capability, low communication bandwidth and limited battery power.

The limited capabilities of IoT devices often mean they cannot afford complex cryptography algorithms due to their computation cost, energy consumption (mostly battery-powered), and limited storage [40]. Moreover, as these devices are resource-deficient, the Internet Security Protocol (IPsec) at the network level may not efficiently function because the negotiation is computationally heavy and the data overhead is high [41]. Furthermore, other conventional cryptography approaches to preserve data integrity such as Advance Encryption Standard (AES) are inadequate too. Therefore, lightweight solutions, either IDS, firewall or cryptographic, should be deployed on resource-constrained devices.

2.1.3.4 Privacy

IoT devices generate a huge amount of data that can be further analysed and processed for decision-making. There are many applications such as smart surveillance cameras and patient monitoring devices where penetrating the sensitive recorded information may lead to dangerous consequences. These pieces of personal information are sometimes revealed to the outside world and may be exposed by intruders. Mutual authentication protocols, strong (yet lightweight) cryptography and effective access control policies are required to preserve privacy in the dynamic and large-scale constrained IoT system. As stated by [42], we do not yet have comprehensive standard privacy policies that specifically address the IoT paradigm.

2.1.3.5 Large scale/scalability

The interactions between heterogeneous devices complicate possible security solutions and deployments. Moreover, IoT ecosystems are scalable by their nature; thus there is a need for scalable and resilient security solutions. In some IoT applications, updating an enormous number of devices cannot be performed in a short time due to large-scale and incompatibility issues [36]. Another challenge is regarding “automatic control.” Most IoT nodes are not controlled and supervised by humans. Therefore, they need to involve self-configuring, self-optimising, self-management, self-healing and self-protecting mechanisms.

2.1.3.6 Trust management

Preserving trust relationships between nodes in IoT ecosystem is still challenging. Cross-layer trust is required to achieve user acceptance of IoT services and applications. Gaining trust is based on the stability of identity management and access control mechanisms [43]. In addition, the associations and connections between IoT devices are changing; they may be replaced or moved (i.e., mobility aspect) over time. This makes the design of trusted solutions complicated.

2.1.4 Security goals

As aforementioned, the beauty of the IoT era is that exchanged information can be accessed anywhere at any time. Moreover, smart devices can interact with each other with minimal human intervention. In fact, they provide new applications and services to end users such as smart homes, smart transportation, etc. However, for these smart nodes to provide continued benefits, they must meet and consider the five pillars of Information Assurance (IA). These are now addressed below.

2.1.4.1 Confidentiality

Keeping transmitted information confidential is crucial for the Internet of Things ecosystem. It means only authorised parties can disclose the data. Access to information must be restricted to those who are eligible to view it. Adversaries can discover the communicated data and violate data confidentiality [44]. Thus, transmission should be secured and encrypted for the IoT network.

2.1.4.2 Integrity

The message in transit (or even at rest) must be secured from being altered or modified. The data has to reach the receiver device in the same condition as it was sent from the sender node. Integrity ensures that an attacker or unauthorised party has not changed the information while in transmission.

2.1.4.3 Availability

Availability is all about providing data or services as needed. The resources and information should not be unavailable whenever nodes or users request. In the IoT ecosystem, nodes are talking to each other. They need each other's data to satisfy the main goal of the IoT paradigm. Attackers can affect the availability of devices and servers by overwhelming the network bandwidth with significantly increased network traffic making resources unavailable upon request. Such attacks that compromise availability are denial of service (DoS), flooding, and jamming attacks.

2.1.4.4 Authenticity

To ensure that only authenticated parties are allowed to access specific resources, authenticity requires proof of identity. Nodes should be able to authenticate each other to communicate. The authentication process is invoked to ensure access is restricted to identified legitimate parties.

2.1.4.5 Non-Repudiation

Generally, non-repudiation means ensuring that a sender and receiver cannot deny what they send or receive. Therefore, nodes in IoT networks should not be able to repudiate packets or data they have previously initiated (e.g., via clone-ID/sybil attack).

2.1.5 The need for an efficient routing protocol

The Internet Engineering Task Force (IETF) created the IPv6 low-power wireless personal area networks (6LoWPAN) as an adaption layer (as shown in Figure 2.1) to allow sensor nodes to implement the Internet protocol (IP) stack and become accessible by other devices on the network. These adaption layers enable these nodes to implement routing protocols at the network layer and provide end-to-end connection, allowing for a wide range of applications. With the Internet's exponential expansion and the emergence of the vast number of resource-constrained IoT nodes, traditional routing protocols can no longer be used. As a result, the RPL was standardised specifically for the IoT-constrained environment and recently gained favour in the research community. Currently, many researchers accept it as the routing protocol for the IoT [45].

2.2 RPL

The Routing Protocol for Low Power and Lossy Networks (RPL) creates efficient routes between the devices to preserve their resources [46]. In LLNs networks, the nodes are resource-constrained in terms of memory and processing and they usually operate using a battery-enabled energy source. The communication between these devices incurs a low packet delivery ratio (PDR) and high packet loss (or dropped packets) due to a high collision rate. In general, LLNs that use RPL as a routing protocol inherit two features: a meagre data rate and very high collision/dropping packet rates, which negatively impact the application throughput.

The RPL-based network uses a distance vector routing mechanism to construct a hierarchical structure in a tree topology called Destination Oriented Directed Acyclic Graph (DODAG). The RPL creates the Directed Acyclic Graph (DAG) based on the ranks assigned to the nodes. These ranks are imposed by the IPv6 Boarder Router (6BR). The ranks are calculated using different methods such as Link Quality (LQ), Expected Transmission Count (ETX) and Hop-Count. The 6BR usually has a rank of 1 and the nodes in the DAG are assigned ranks based on their distance from the 6BR. The closer the nodes are to the 6BR, the lower the rank and the more valuable they are. Nodes that are close to the 6BR become parents to other nodes down the tree. An example of a DAG is illustrated in Figure 2.2.

The DODAG is created based on several exchanged control messages as follows:

- **DODAG Information Object (DIO)**: during the RPL network initialisation, the 6BR or root node starts the process by multicasting the DIO messages to all its neighbouring nodes. The DIO includes important information such as the rank, RPL instance, parent list, siblings list and others. Based on the rank, a receiver node will choose a parent node, in this case, the 6BR. Then, nodes that are in radio range of the 6BR and have received its DIOs, become parent nodes to other nodes by advertising their new calculated ranks in their DIO messages.

- **Destination Advertisement Object (DAO)**: The receivers of the DIO packets respond with unicasted DAO messages to confirm the sender as a parent node.

- **DAO-acknowledgement (DAO-ACK)**: The parent nodes reply with DAO-ACKs to each child node as an acknowledgement of parenthood.

- **DODAG Information Solicitation (DIS)**: For new nodes to join the DAG, they

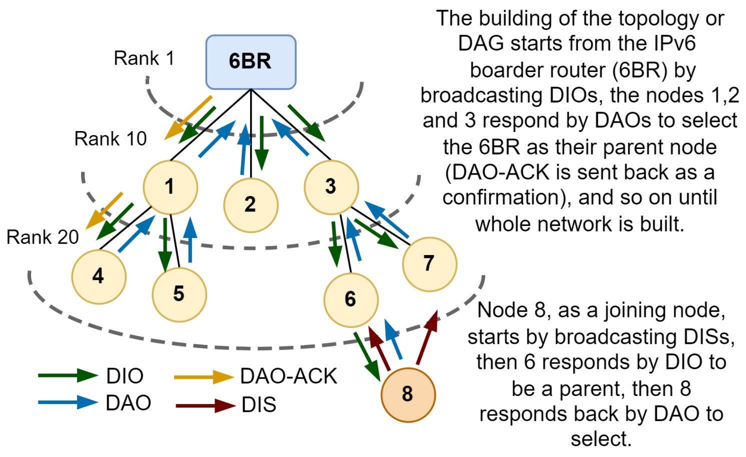

Rank 1
6BR
Rank 10
Rank 20
1
2
3
4
5
6
7
8
The building of the topology or DAG starts from the IPv6 boarder router (6BR) by broadcasting DIOs, the nodes 1,2 and 3 respond by DAOs to select the 6BR as their parent node (DAO-ACK is sent back as a confirmation), and so on until whole network is built.
Node 8, as a joining node, starts by broadcasting DISs, then 6 responds by DIO to be a parent, then 8 responds back by DAO to select.
DIO
DAO
DAO-ACK
DIS

Figure 2.2: *RPL DAG*

need to multicast DIS messages to discover the neighbourhood for any DIO messages, i.e., a possible parent node.

The process of exchanging these control messages continues until the DAG is formed and completed.

2.2.1 Threats to RPL-based Internet of Things

RPL is the most common and standardised routing protocol for IoT networks [11]. However, the network is vulnerable for two fundamental reasons. First, the RPL protocol standards do not provide a routing security mechanism when configuring the network [47]. Thus, malicious nodes can carry out their activities while packets are being routed, allowing various types of attacks on the routed data. Second, the nodes are not tamper-proof and are easy to compromise [48]. The previously applied countermeasures to the traditional IP-enabled networks may not work well in RPL-based IoT networks for several reasons: the computation capabilities for the devices or nodes are limited, the power supply is restricted, a vast number of devices are interconnected via unreliable and lossy connections, and the storage or memory size of these constrained devices is insufficient. These limitations make the RPL networks more vulnerable than other conventional networks.

Figure 2.3 presents a taxonomy of the most common attacks. We can see that the ad-

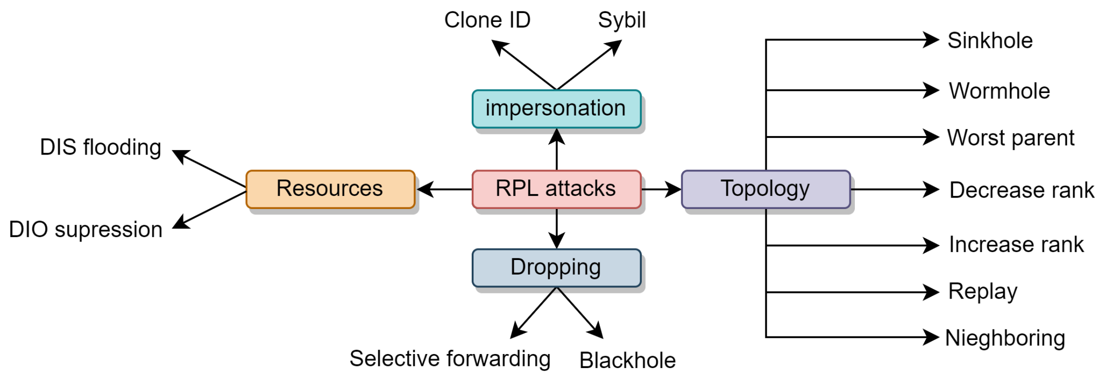


Figure 2.3: *A taxonomy of the most common RPL attacks*

versary could target the nodes' resources by draining their energy, the network topology by creating an unoptimised routing path, the nodes' ID by impersonating them, or the communication by dropping application packets. Consequently, integrity, confidentiality, availability and non-repudiation are at risk. Next, we detail the common attacks targeting RPL networks.

2.2.1.1 Sinkhole, Blackhole and Selective forwarding

In the sinkhole attack, the attacker node advertises a fake rank when multicasting the DIO messages. This rank is higher than that of the neighbouring nodes to mislead the nodes in the neighbourhood to select it as a parent node. This attack affects the optimised path created by the RPL where the child nodes of the attacker nodes choose it as a preferred parent to deliver their packets. As shown in Figure 2.4, the blackhole takes a step further and drops all received packets. On the other hand, the selective forwarding attack drops some of the received packets and forwards others. The selection of which to forward or drop is based on either the sender ID, the packet type or just random. Algorithm 1 demonstrates the process of these attacks.

2.2.1.2 DIS Flooding

The aim of this attack is to increase the network overhead by generating a huge number of exchanged control messages. As presented in Figure 2.5 and Algorithm 2, the DIS-based attacker node overwhelms the network with many DIS control messages. It continuously

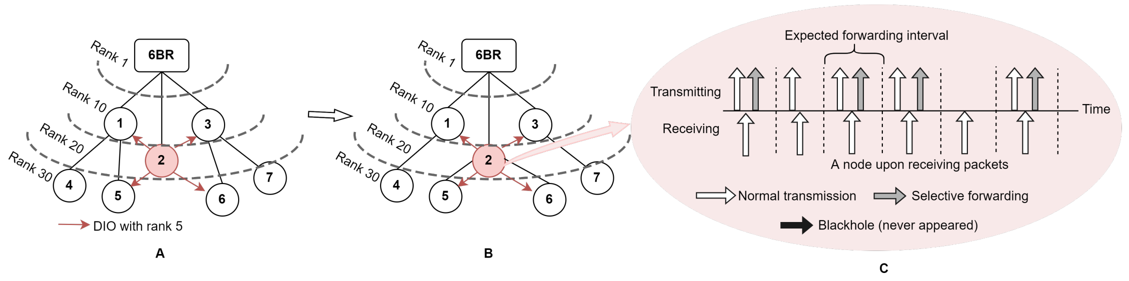


Figure 2.4: *Sinkhole, blackhole and selective forwarding attacks*

Algorithm 1: Sinkhole, Black-hole, Selective_forwarding attacks

```
1 Initialisation
2 A: Attacker node
3 N: Neighbour list ⊂ legitimate LLN nodes
4 B: a neighboring node ∈ N
5 P: Current packet
6 R: a lower more powerful rank, usually assigned as the sink node rank
7 Attack_type = {Sinkhole, Black-hole, Selective_forwarding}
  Input: "A" receives DIOs from A.N and calculates Min(advertised_ranks)
  Output: "A" obtains a lower, malicious rank and multi-casts it with DIO to all
          nodes, ∀ node ∈ A.N
  if (P is DIO) ∧ (P.sender_id ∈ A.N) then
      if P.sender_id ∈ A.N ∧ P.sender_id ≠ root_id then
          if DIO.rank ≤ A.malicious_rank then
              A.malicious_rank ⟵ R
  if (A.received(DIS from B)) ∨ (A trickle_timer activated) then
      A.multicast((DIO with malicious_rank) to node ∀nodes ∈ A.N)
      B.receive(DIO from A)
      if DIO.rank < B.rank then
          B nominate A as preferred_parent
      B unicast application packets to its preferred_parent, which is "A" now, in order
       to transfer it to the destination
      if Attack_type is Sinkhole then
          A collect packets from B and transfer it to next hop
      else if Attack_type is Blackhole then
          A collect packets from B then drop all of them
      else if Attack_type is Selective_forwarding then
          A collect packets from B and selectively or randomly drop some and transfer
           others to next hop
```

broadcasts DIS packets to all its neighbours (in case of a multicast DIS flooding) or to a specific neighbour node (in case of a unicast DIS flooding). This affects the resources of the neighbouring nodes as they need to reply with DIO messages every time they receive DIS packets. As mentioned in Section 2.2, the DIS control messages are only used by either a node that lost its connection or a new node that wants to join the network. They use the DIS to discover the area searching for a DODAG to connect to.

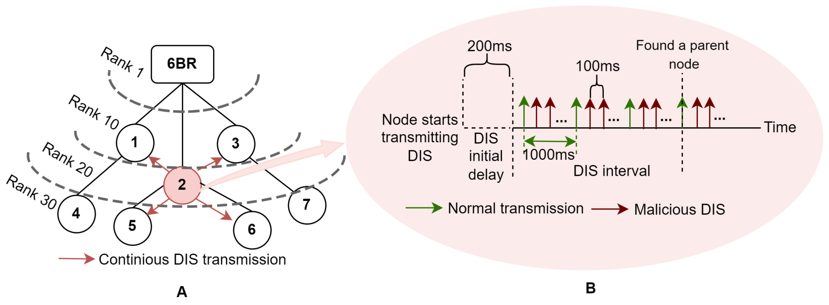


Figure 2.5: *DIS flooding attack*

Algorithm 2: DIS Flooding attack

```
1 Initialisation
2 A: Attacker node
3 N: Neighbour list
4 I: Current node id
5 B: a neighboring node $\in N$
6 V: Victim list
7 P: Current_packet
8 Attack_type = {Unicast DIS Flooding, Multicast DIS Flooding}
9 Control_Packet = {DIO, DAO, DIS, DAO-Ack}
  Input: "A" uni-casts or multi-casts DIS to node(s), $\forall$ nodes $\in A.N$
  Output: "B" uni-casts or multi-casts DIO message
  if A.Attack_type is Unicast DIS Flooding then
      A.unicast(DIS $\Longrightarrow$ B, $B \in A.N$)
      B.unicast(DIO $\Longrightarrow$ A)
  else if A.Attack_type is Multicast DIS Flooding then
      A.Multicast(DIS, $\forall B \in A.N$)
      for $\forall B \in A.N$ do
          B.Multicast(DIO, $\forall node \in B.N$)
```

2.2.1.3 Increase Rank

This attack targets the resources of LLN nodes, causing victim nodes to run out of energy. It also disrupts the communication within the LLN. The intruder starts the attack, as shown in Figure 2.6 and illustrated in Algorithm 3, by raising its rank and multicasting this fake rank (less valuable) to its neighbouring nodes via the DIO messages. This forces the children to look for a new parent to reach the border router (6BR). When the children locate a new parent after generating a significant network overhead, the adversary node returns back to the old rank or broadcasts a lower (better) rank to entice the neighbours to re-nominate it as a parent.

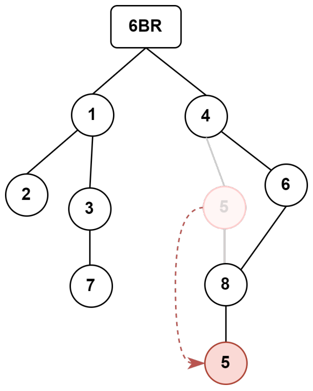


Figure 2.6: *Increase rank attack*

2.2.1.4 Wormhole

The adversary nodes (usually more than one node) create an unoptimised route to the 6BR for the neighbouring nodes. Two or more distributed wormhole nodes create a private tunnel between them to pass in the collected packets/information from each part. In this kind of attack, the attacker node attracts the neighbouring nodes by advertising a wider radio range capability (i.e., lower rank). The collected packets from one part of the network are then sent to the other wormhole node via the private channel. Figure 2.7 and Algorithm 4 present a demonstration of such an attack.

Algorithm 3: Increase rank attack

```
1 Initialisation
2 A: Attacker node. N: Neighbour list. P: Current packet. $R_1$: is the initial, legitimate
  rank. $R_2$: is a high, less valuable rank.
Input: "A" increases its rank to much higher rank and multi-casts it with DIO
Output: "A" receives a DIO containing a lower rank from a neighbouring node,
        then decreases its rank and multi-casts it with DIO
if $(P = DIO) \wedge (P.sender\_id = A.id)$ then
    if $A.rank = R_1$ then
        $A.rank \Longleftarrow R_2$
    else if $A.rank \neq R_1$ then
        $A.rank \Longleftarrow R_1$
    A.Multicast(DIO to A.N)
if $(P$ is $DIO) \wedge (P.sender_{id} \in A.N)$ then
    if $(P.Rank < A.Rank) \wedge (P.sender\_id \neq rootnode\_id)$ then
        $A.rank \Longleftarrow R_1$
        A.Multicast(DIO, $\forall$ nodes $\in$ A.N )
```

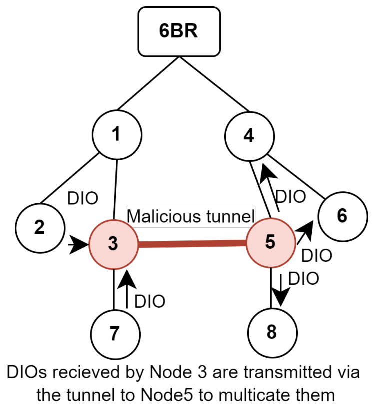

6BR
1
4
2
DIO
DIO
6
Malicious tunnel
3
5
DIO
DIO
DIO
7
8
DIOs recieved by Node 3 are transmitted via the tunnel to Node5 to multicate them

Figure 2.7: *Wormhole attack*

2.2.1.5 Worst parent

As the name indicates, the intruder selects the worst possible parent rather than the one with the optimal (lowest) rank. This means the route of the packets will go through a long path to reach the border router (6BR). An illustration of this attack is given in Figure 2.8 and Algorithm 5. This attack affects not only the compromised node, but also the children nodes of it, especially if it was in a strategic position with many children nodes.

Algorithm 4: : Wormhole attack

```
1 Initialisation.
2 $A_1, A_2$ :Attacker 1 and 2.  N: Neighbour list.  B: a neighboring node $\in$ N.  P:
    Current packet
3 Control_Packet = {DIO, DAO, DIS, DAO-Ack}
  Input: B Multi-casts Control_Packet to nodes, $\forall$ node $\in$ B.N
  Output: $A_1 or A_2$ transfer the received Control_Packet from B to its counterpart
  if (P is Control_Packet) then
      if $P.source\_id \in A_1.N$ then
          $A_1$.transfer($P$ to $A_2$)
          $A_2$.multicast($P$, $\forall$ node $\in A_2$.N)
      else if $P.source\_id \in A_2.N$ then
          $A_2$.transfer($P$ to $A_1$)
          $A_1$.multicast($P$, $\forall$ node $\in A_1$.N)
```

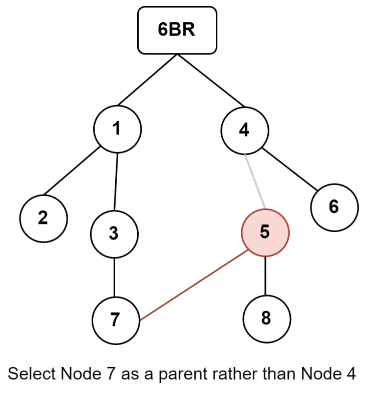


Figure 2.8: *Worst parent attack*

Algorithm 5: Worst Parent Attack

```
1 Initialisation
2 A: Attacker node
3 N: Neighbour list
4 P: Current packet
5 Control_Packet = {DIO, DIS, DAO, DAO-Ack }
  Input: "A" discovers the neighbouring node with the highest, least valuable rank,
         providing the worst OF
  Output: "A" selects the discovered worst parent as the preferred parent to reduce
          routing performance
  if (P = DIO) $\wedge$ (P.sender_id $\in$ A.N) then
      if (A.preferred_parent[rank] < Node.Rank, $\forall$ Nodes $\in$ A.N) then
          A.preferred_parent = Node.id
```

2.2.1.6 DIO suppression

The DIO suppression attack aims to slow down the transmission of DIO packets in the network. To make this happen, the attacker exploits the Trickle mechanism as illustrated in Figure 2.9. The intruder sends DIO messages continuously that are considered consistent by the neighbouring nodes. After receiving enough of the same DIO messages, neighbouring nodes suppress their own DIO messages [1]. The DIO, in general, is required to inform the internal nodes about the DODAG information, as described in Section 2.2. However, this continuous suppression leads to some nodes and routes not being discovered.

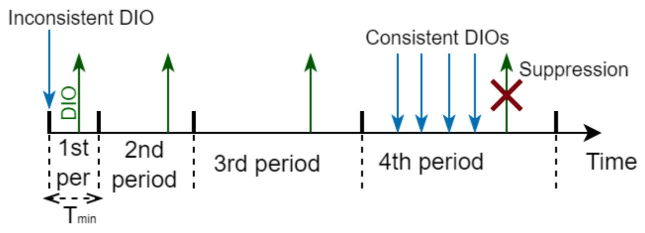


Figure 2.9: *DIO suppression [1]*

2.2.1.7 Sybil attack

The Sybil attack in the conventional internet is when an adversary controls several accounts (e.g. on Twitter) to perform malicious activities. Similarly. In the RPL network, the Sybil adversary clones identities (MAC or IP address, rank,...etc.) of several victim nodes to function as them. The same malicious node operates with multiple identities, as demonstrated in Algorithm 6. Later, the attacker node multicasts or unicasts the control messages of those cloned nodes.

Algorithm 6: Sybil Attack

```
1 Initialisation
2 A: Attacker node.
3 N: Neighbour list.
4 P: Current packet.
5 L: Target List.
6 S: sender node
7 Control_Packet = {DIO, DIS }
8 Attack_Types = {Sybil, Clone_Id}
Input: Control_packet initiated by victim node(s)
Output: "A" steals victim nodes credentials, then uni-casts or multi-casts control
        packets with their identities
if $(P \in Control\_Packet) \wedge (P.sender\_id \in A.N)$ then
    //attacker can select the victim(s) selectively or target its children
    if $(S.node\_id \in A.children\_list)$ then
        if $(Attack\_Type = Sybil)$ then
            if $(S.node\_id \in A.L)$ then
                $A.clone(S.credential)$
            $A.Multicast(Control\_Packet \Longrightarrow A.N) \vee A.Unicast(Control\_Packet \Longrightarrow A.N[node\_id])$
        else if $(L.length = 1 \wedge Attack\_Type = Clone\_Id)$ then
            $A.clone(S.credential)$
            $A.Multicast(Control\_Packet \Longrightarrow A.N) \vee A.Unicast(Control\_Packet \Longrightarrow A.N[node\_id])$
```

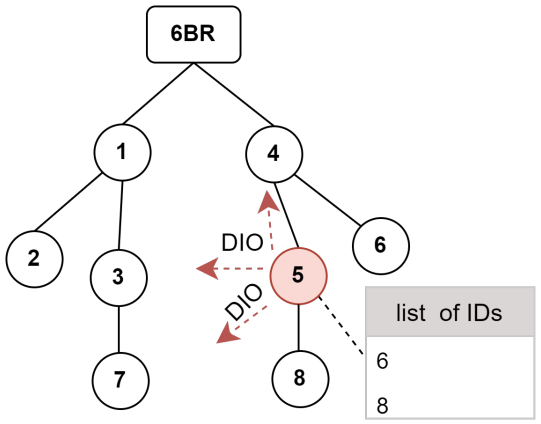


Figure 2.10: *Sybil attack*

2.2.1.8 Replay attack

The intruder collects some control packets (such as DIO, DAO and DIS) from its neighbouring nodes. Subsequently, it distributes these recorded messages among the neighbouring nodes. The process is given in Algorithm 7. These packets contain old information and multicasting them could cause unoptimised routes and wrong node information such as parent and sibling lists. A similar attack is called a *neighbour attack* (or DIO replay attack) where the malicious node multicasts the received DIO messages (without update) to its neighbouring nodes [49]. They consider these nodes, the sender of these DIO messages, as their neighbours (while they are not). Consequently, they may update their routes to the out-of-range neighbours. This leads to creating wrong routes, disrupting the DAG topology, or consuming more energy.

Algorithm 7: Replay Attack

```
1 Initialisation
2 A: Attacker node N: Neighbour list P: Current packet L: Target List R: List of
    recorded control packets
3 Control_Packet = {DIO, DIS, DAO, DAO-Ack }
  Input: "A" records Control_Packet initiated by "L"
  Output: "A" multi-casts R
  if (P ∈ Control_Packet) ∧ (P.sender_id ∈ A.N) then
      //attacker can select the victim(s) selectively or target its children
      if (P.sender_id ∈ A.L) then
          Add(R ⟸ P)
      if (Attack_triggertimer.status = Activated) then
          A.multicast(R, ∀ nodes ∈ A.N)
```

2.2.2 RPL-based IoT security vulnerability

The IoT infrastructure involves four layers. Compromising any one of them may have a significant impact on the others. It is essential to consider security at each layer: security measures at *the physical layer* to ensure the confidentiality and integrity of the gathered data; security measures at *the network layer* to protect the transmitted data; Security measures at *the processing layer* to provide trustworthy decision-making about the supplied data; and security measures at *the application layer* to maintain secure and beneficial operation.

Focusing on the *network layer* where the routing protocol lies, community researchers have proposed many projects to enhance the security side of the RPL-based IoT network. This is to provide strong defence mechanisms to maintain data confidentiality, integrity and availability. Some of these defence mechanisms are cryptography [50], authentication [51], and trust-based [52]. The architecture has to be highly secured and privacy should be preserved to gain users' satisfaction to deploy and utilise the IoT applications.

However, because the RPL nodes are resource-deficient and lack tamper-resistance [48], intruders may find novel ways of penetrating provided defence mechanisms and countermeasures. Therefore, the RPL-based IoT networks require another line of defence to provide continuous protection. Intrusion Detection Systems (IDSs) exist to serve such a purpose.

2.3 Intrusion Detection Systems in the IoT/RPL

An Intrusion Detection System (IDSs) are tools or software to detect unauthorised access to a network or system. IDSs have been around for over three decades to efficiently provide security solutions for traditional IP networks. The concept was first introduced in 1980 by James P. Anderson [6], a pioneer in information security and regarded by many as the founder of the Intrusion Detection System field. IDSs gather information about a system or network to execute a security status diagnosis. The aim is to identify security breaches or open vulnerabilities that might result in breaches.

The IDS can detect both internal and external attacks. Internal intrusions are launched on the network by compromised nodes that are part of a network. On the other hand, external intrusions are initiated by third parties from outside the network. As depicted in Figure 2.11, there are three main components of most IDSs [53]. The first module is the *IDS Sensor or Data Collection* module. It is responsible for monitoring the network and collecting data. Generally, sensing capability can be distributed (DIDS) on the network or centralised (CIDS) at a specific collection point. The second module is the *Detector* engine where the data collected is then analysed to detect any unauthorised activities. It uses the knowledge, whether this is previous attacks' fingerprints or anomaly training, to make a

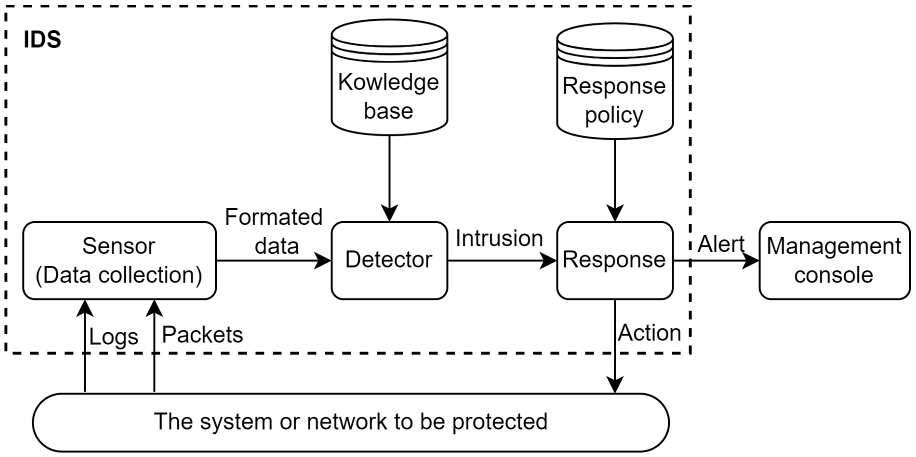


Figure 2.11: *IDS components*

decision. The *Response* module is the third component. It generates an alert or else takes an action concerning malicious activities. If the latter, then this is called an Intrusion Prevention System (IPS), which can function alongside the IDS.

The IDS is an effective extra line of defence when other security measures, such as authentication, access control and others, fail to protect the target assets. However, the constrained nature of many IoT network devices and nodes makes the deployment of IDSs difficult and sometimes impossible. Devices integrated into IoT networks have limited processing capability, low storage capacity, and lower battery life. Therefore, hosting an IDS agent (e.g., ML-based) on these constrained nodes is sometimes inapplicable, or else it may affect the fundamental operational function of these IoT devices (e.g., sensing the environment) [54]. Another issue is related to the communication techniques of IoT networks. Unlike traditional networks where nodes communicate and are connected to a specific router or access point to forward packets, nodes in IoT mostly communicate hop-by-hop and each node can both forward messages and act as an end-system. Therefore, there is a need for in-network distributed monitoring. Another difficulty is related to the protocols used explicitly for IoT-constrained networks, which are still in their infancy. Some of the protocols are IEEE 802.15.4, RPL, 6LoWPAN, Constrained Application Protocol (CoAP),and Message Queuing Telemetry Transport (MQTT). Each of these protocols has its specific vulnerabilities and requires a specially designed IDS.

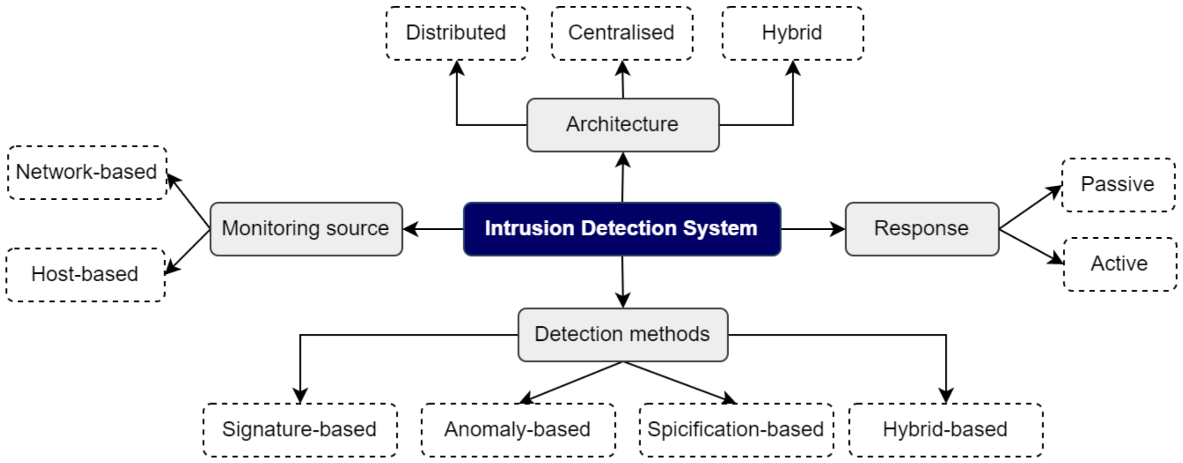


Figure 2.12: *IDS classification*

The following section provides an overview of several techniques integrating the IDS into the IoT/RPL network. All reviewed IDS proposals are summarised in Table 2.3. The structure of any RPL-IDS is classified into four main categories: monitoring source, deployment architecture, detection methods, and response. Intrusion detection taxonomy is demonstrated in Figure 2.12.

2.3.1 Monitoring source

IDSs are classified, based on the data source they monitor, into Host-based IDSs (HIDSs) and Network-based IDSs (NIDSs) [55].

- In the HIDS approach, the detection system is installed in a device or node to protect the system from any malicious activities. It monitors the audit trails of system logs and calls, file system changes and application logs.
- The NIDS, on the other hand, captures and analyses the network packets looking for intrusions. It is either connected to intersection points of the network traffic, such as routers, or distributed over the network.

2.3.2 Intrusion detection architectures

The system architecture can be classified based on where the monitoring module (i.e., the probe) and analysis module (i.e., decision-making) are located. The placement strategy for the IDS modules can be divided into three categories: distributed, centralised, and hybrid. Next, we detail each one of them.

2.3.2.1 Distributed IDS (DIDS)

In this architecture, distributed monitor nodes hold all IDS components such as sniffing, analysis and reporting modules. The detection agent could be placed at different locations (either on network's sensor nodes themselves or on separate stand-alone nodes) in the IoT network. Every monitoring node is responsible for monitoring itself and the neighbour nodes for possible attacks. The decision on an intrusion could be taken locally. As such, the communication overhead might be decreased, yet, it requires more resources such as storage, processing and energy for each node. Therefore, detection modules need to be optimised to reduce the resources usage while maintaining high detection rate. Figure 2.13 presents an example of such deployment.

Cervantes et al. [56] propose an approach targeting the detection of Sinkhole attacks on 6LoWPAN where each node is responsible for monitoring its neighbours (i.e., in a watchdog mode). When a node detects an attack, it broadcasts a message to alert the other nodes and isolates the compromised node. This approach organises the network into clusters to maintain scalability. The network's nodes are divided into free nodes (not belonging to any cluster and can move around), member nodes and leaders. However, all nodes including leaders are responsible for monitoring neighbours. Their proposal aims to address the mobility issue of IoT networks.

2.3.2.2 Centralised IDS (CIDS)

In this strategy, there are two approaches. The first approach is when the information-gathering modules are distributed on the network for either system monitoring or network

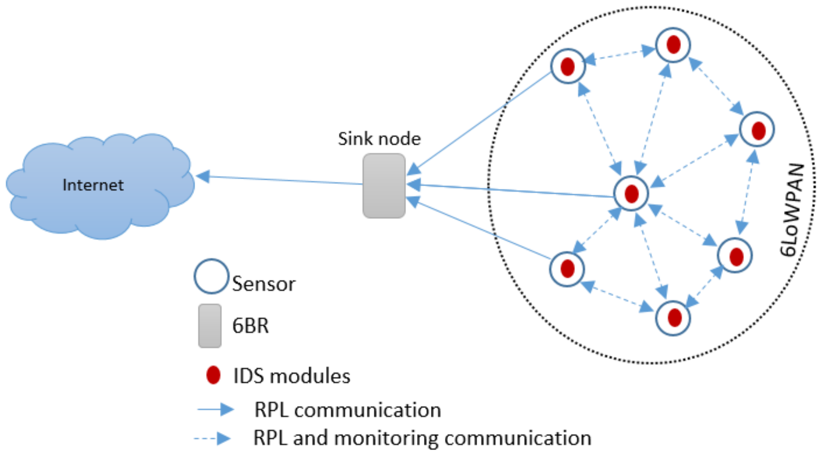


Figure 2.13: *Typical distributed IDS*

packet inspection. The analysis engine is hosted at a centralised powerful device for either carrying out correlations or mining further information. In RPL-based IoT networks, analysis modules are usually placed in the border router, while the sniffing modules are hosted either in the IoT sensors themselves or in separate sensor nodes. It may create a communication overhead as these monitor nodes need to send network or neighbour status periodically to a centralised analysis device.

This technique is used by [57] where they propose an IDS framework for IoT that has a monitoring system and detecting engine. It is targeted at the 6LoWPAN networks. They introduce IDS probes to sniff packets and provide information to the IDS. They adopt the Suricata (an open-source IDS), but since it cannot support 6LoWPAN protocols, they implement a crafted decoder to inspect the incoming packets. The Suricata works as a detection model and Prelude (Event monitoring system/Security information and event management (SIEM)) as a management module.

The second centralised approach places the IDS (both monitor and analysis units) in the border router (i.e., 6BR) where all network traffic (those going to the global/external networks) passes. An example of such a topology is illustrated in Figure 2.14. This approach could eliminate the communication overhead and allow the IDS to inspect and analyse the network traffic travelling between internal nodes and the global network. Nevertheless, the multi-hop aspect of the RPL-based network allows the nodes to communicate internally without border router knowledge. Thus, a compromised node could spread malware or nasty packets inside the network. In the research presented in [58], the IDS instance is placed on the edge. They propose the integration of Complex Event Processing (CEP) with the detection module to improve the real-time detection of malicious traffic. The authors deploy the IDS processing at the edge since it offers low response time, incurs reduced bandwidth and provides high computation capabilities.

For both approaches and since the heavy-duty work is centralised, traffic inspection for each node will be reduced and processing capacity will be increased. Yet it represents a single point of failure.

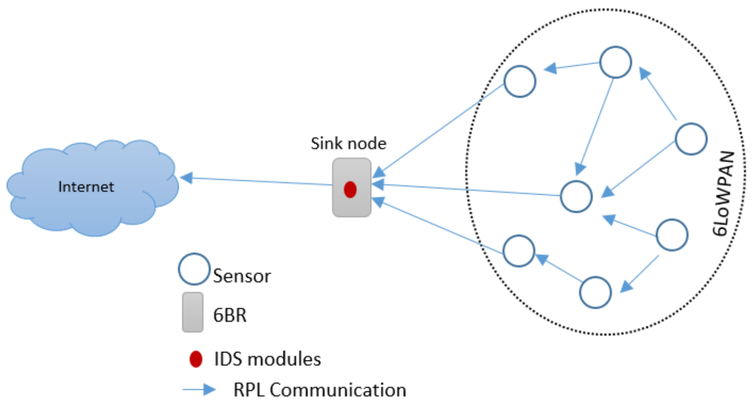


Figure 2.14: *Typical centralised IDS*

2.3.2.3 Hybrid IDS (HIDS)

This type of deployment combines the two previously discussed strategies. It seeks the best of both approaches. In the hybrid deployment architecture, there are two approaches that have been discussed in the literature.

The first is when the network is divided into two or more groups or clusters. One dedicated node in each cluster hosts the IDS agent to monitor other nodes in its group. This node is called a cluster head and is usually resourced with more energy and processing capabilities. Each member node periodically sends information about itself and its neighbours to the cluster head. Figure 2.15 shows an example of such deployment. A watchdog or leader node has a set of detection techniques to decide whether the nodes in its region are compromised or not. According to [59], a leader node is often more robust. The authors in [60] have adopted this approach. In order to maintain the resources of the cluster heads, they make the members send their status to cluster heads at each time interval rather than having them overhear all network traffic.

Another way of adopting the hybrid deployment approach is by having different lightweight detection modules on different nodes (i.e. probes) for monitoring/detection purposes and dedicating a more resourced node or nodes (i.e. a centralised analysis unit) for further detection. The distributed modules are able to make local decisions. Yet, they might need the centralised units for correlation and aggregation purposes. In an IoT-based IDS proposal called SVELTE [8], the IDS functionality is hosted in both the border router (6BR) and each network node. It has a sense of centralisation where the intensive work, like intrusion detection by analysing gathered data, is at the border router (6BR). The other nodes hold lightweight detection tasks like sending routing-based data to the border router and informing it about any malicious traffic they notice. Another research that embraced this method is proposed by [61]; they deploy two modules: local and global. In the local stage, detection sniffers (DS) are distributed, in promiscuous mode, to classify the network traffic (packet counts) based on a decision tree classifier to generate Correctly Classified Instances (CCIs). When the calculated CCI is sent by a DS to the Super Node (SN), the global stage collects CCIs and applies Iterative Linear Regression to create a time-based profile and threshold to differentiate between malicious nodes and normal ones.

2.3.3 Detection methodologies

There are four intrusion detection methods: signature-based, behaviour-based, specification-based (also known as rule-based) and hybrid-based. Next we detail each one of them.

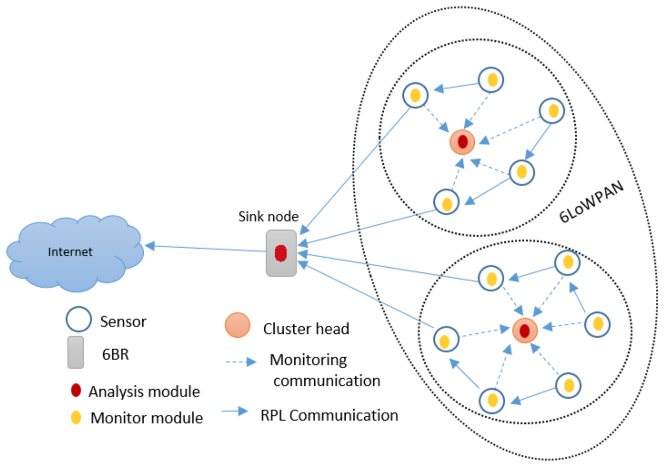


Figure 2.15: *Typical hybrid IDS*

2.3.3.1 Signature-based approach

In the signature-based, also known as misuse-based, approach, the IDSs can detect attacks based on their previous signatures stored in a database. Essentially, a signature-based detector looks for known patterns (fingerprints) of malicious behaviours. This technique is very accurate and can effectively detect known intrusions. However, attacks change and new attacks emerge rapidly over time and this approach can not generally detect previously unseen threats (i.e., zero-day attacks). Furthermore, the number of attacks signatures stored in a database will increase over time requiring more storage capacity [62], which is usually not available in the RPL-based IoT networks. The misuse-based method has been used in [63] to protect IoT devices against internal and external attacks especially DoS types of attacks. They use the Cooja simulator for their experiments and target the RPL protocol. They conclude that these attacks affect IoT devices' availability and power consumption.

2.3.3.2 Anomaly-based approach

Unlike the signature-based approach, the Anomaly-based method, also known as Behaviour-based, has the ability to detect some new threats. It compares traffic and activities with the profile of 'normal' behaviour. If system activities or network behaviour exceed a predefined normal threshold, it generates an alert indicating suspicious activity. However, this approach considers all activities that do not match normal behaviour as intrusions to the system. This may raise the false positives rate [53]. The idea of the intrusion detection technique proposed by [64] is to look for any contradiction in the network by monitoring the specification of 1-hop communication such as packet size and data rate. The system then can distinguish normal from abnormal behaviour. Furthermore, they propose an internal anomaly detection mechanism integrated with RPL (the routing protocol). They utilise a Distress Propagation Object (DPO) control message to report the anomaly to parent nodes and then to the edge-router. The latter performs periodic consistency checks. It processes the received information from nodes to make a decision and issue a report to users. The system involves three modules: monitoring, reporting and isolating.

2.3.3.3 Specification-based approach

The term specification-based refers to a set of rules (with thresholds) applied to distinguish network or system normal activities and malicious ones based on the behaviour of the network/system when it deviates from specified behaviour. This method is sometimes referred to as rule-based IDS. Thus, it is an anomaly-based approach that raises an alarm whenever network behaviour deviates from a specified normal state. Network administrators mostly define these specifications (protocols) manually [62]. Alternatively, these rules can be generated or evolved using Artificial Intelligence (AI) techniques such as Genetic Programming (GP) and Decision Trees (DTs). This approach is claimed to have lower false positive rates and be less computation-hungry than anomaly-based approaches.

Le et al. [60] utilise a specification-based IDS to detect topology-based attacks in RPL-based networks. A set of specifications of RPL are obtained based on profiling the nodes' behaviours and are deployed (as rules) on the IDS agents. The work was motivated by the authors' assessment that anomaly detection is not feasible for topology-based attacks. They divided the network into clusters where the head of each cluster hosts an IDS. Based on a set of implemented rules, cluster heads monitor members in their clusters. The members are required to report their current state to their cluster head at each time interval. The authors involve Finite State Machines in profiling RPL operation. During the simulation, they run the network normally and record the state and transitions in a trace file.

2.3.3.4 Hybrid-based approach

The fourth approach aims to achieve the best outcomes and minimise the drawbacks of signature and anomaly-based approaches [53]. One of the most effective hybrid approaches for IoT networks is Kalis [65]. This knowledge-based system is able to detect attacks in real time. Knowledge is shared between the network nodes to make better decisions. The design does not require changing the IoT nodes' software since it can be deployed in a separate device as a standalone tool. It can work with multiple protocols and the performance overhead is reduced compared to other traditional IDSs and Snort [66]. Another relevant proposal is SVELTE[8]; it uses a hybrid detection method and hybrid placement strategy as stated earlier.

More related works and details are presented in Section 2.3.6 and Table 2.3.

2.3.4 Response

The IDS recognises malicious activities or violations and provides alerts to the network administrator. Intrusion detection reaction could be either passive or active.

- In passive responses, the IDS logs abnormal activities and alerts a user or network administrator about the detected behaviour.
- An active IDS, in addition to logging and alerting, may initiate specific countermeasures against perceived suspicious or malicious behaviour, for example by arranging for the blocking of a specific node's communications. This is known as an Intrusion Prevention System (IPS).

2.3.5 Intrusion detection performance metrics

There have been many IDSs developed by either industry or the research community. A confusion matrix, also known as an error matrix, is often used to evaluate their performance.

It represents, in a tabular format, how many predictions a classifier algorithm made correctly and incorrectly. It is a technique to effectively derive the functional metrics and compare different kinds of IDSs performance. However, the effectiveness of any IDS is based not only on these functional measures such as detection rate and false positive rate but also on other non-functional metrics such as deployment cost and resource consumption.

2.3.5.1 Functional metrics

The following evaluation metrics, as also illustrated in Table 2.2, are calculated to measure the ability of an IDS to distinguish malicious events from normal activities:

- **True Positive (TP):** when a malicious event is correctly classified as malicious.
- **True Negative (TN):** when a normal event is correctly classified as normal.
- **False Positive (FP):** when a normal event is incorrectly classified as malicious.
- **False Negative (FN):** when a malicious event is incorrectly classified as normal.

Table 2.2: *IDS confusion matrix*

	Normal(Actual)	**Attack**(Actual)
Normal(Predicted)	TN	FN
Attack(Predicted)	FP	TP

The following basic metrics are the commonly used performance measures:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{2.1}$$

$$Detection\ Rate(True\ Positive\ Rate/Recall) = \frac{TP}{TP + FN} \tag{2.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

$$F1score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{2.4}$$

$$False\ Positive\ Rate = \frac{FP}{FP + TN} \tag{2.5}$$

$$False\ Negative\ Rate = \frac{FN}{FN + TP} \tag{2.6}$$

2.3.5.2 Non-functional metrics

Due to the resource-deficiency of the RPL networks, other non-functional metrics need to be considered for the IDS deployment. Some of the non-functional criteria are:

- **Deployment cost:** Any extra IDS sensor placement incurs an extra cost [67]. This cost could be management cost, financial cost or communication cost. So we need to reduce the number of IDS sensor placements while maintaining the functional criteria.
- **Feasibility cost:** The RPL-based IoT networks involve heterogeneous nodes with different capabilities. We need to take into consideration the resources available on the nodes. Some of them may be able to hold a heavyweight detection mechanism while others may not.
- **Coverage:** In the case of a distributed IDS deployment, which is the preferred architecture for the RPL networks [68, 69], nodes are usually used to monitor their neighbouring nodes for any malicious activity. Nodes may be attacked themselves or may go into sleep mode, thus we need the deployment to be resilient. In other words, the monitored nodes need to be monitored by at least two monitoring nodes.

2.3.6 IDS proposals for RPL-based IoT

In this section, we will present some of the common proposals of IDS for IoT/RPL networks. Table 2.3 illustrates the techniques used in each proposal.

Amouri et al. [61] deploy detection sniffers and super-nodes. The detection sniffers are distributed on the network to collect data from neighbour nodes. These probes employ a decision tree mechanism to classify packets being exchanged in their radio range. The classified instances are then sent to the super node. The latter performs linear regression to identify the malicious nodes based on the classified instances received from sniffers.

The work in [57] proposes a framework that has monitoring probes and an analysis engine. They utilise Suricata as their IDS or detection engine and integrate a decoder since Suricata cannot interpret WSN and 6LoWPAN protocols. To protect their IDS probes from DoS and jamming attacks, they connect them via an Ethernet cable. The evaluation is based on physical IoT devices and the PenTest approach using Linux Metasploit to launch DoS attacks, flooding attacks. However, the framework may consume the IoT nodes' resources and degrade the overall performance [49]. Moreover, it cannot detect novel attacks.

The authors in [56] present INTI, a routing-based IDS to detect attacks in 6LoWPAN networks. The main contribution is to detect Sinkhole attacks in a mobile and dynamic environment. Node roles change over time based on the network configuration and attacks. The monitoring node measures the probability of a node being abnormal by counting the incoming and outgoing streams of packets. They use Cooja simulator to evaluate their work. They claim it outperforms SVELTE [8] in achieving low false positive and false negative rates. They did not address the effect of their proposal on nodes with memory and power constraints.

Another work [64] allocates the detection modules between the monitoring nodes (i.e. probes) and the bridge router. Probes work in a cooperative way to watch their neighbours' behaviour. An alert is sent to a centralised analysis unit if an anomaly is detected. The latter correlates alerts from monitoring nodes and makes the final decision.

SVELTE [8] has two main components: a 6LoWPAN mapper (6Mapper) and an IDS. They utilise the Boarder Router (6BR) to hold the analysis engine of the IDS. Distributed lightweight detection modules are hosted in the IoT nodes. 6mapper is used to construct the DODAG tree. It checks for inconsistencies in the network routing state and for the authenticity of each node to avoid Sybil and Clone ID attacks. To protect the constrained nodes from global attacks, they introduce a mini-firewall to each node. They evaluate their proposal using Cooja simulation and test it against Sinkhole and Selective forwarding attacks.

Table 2.3: *IDS deployment techniques.*

Ref	Monitoring source		Detection method				Placement architecture			Evaluation strategy	
	HIDS	NIDS	SG	SP	AN	HD	CIDS	DIDS	HIDS	Functional	Non-functional
[61]	-	✓	-	-	-	✓	-	✓	-	✓	×
[57]	-	✓	✓	-	-	-	✓	-	-	✓	×
[56]	-	✓	-	-	✓	-	-	-	✓	✓	×
[64]	-	✓	-	-	✓	-	-	-	✓	✓	×
[8]	-	✓	-	-	✓	-	-	-	✓	✓	✓ˣ
[65]	-	✓	-	-	-	✓	✓	-	-	✓	✓ˣ
[59]	-	✓	✓	-	-	-	-	-	✓	✓	✓ˣ
[70]	-	✓	-	-	✓	-	-	-	✓	✓	✓ˣ
[71]	-	✓	-	-	-	✓	-	-	✓	✓	✓ˣ
[58]	-	✓	✓	-	-	-	✓	-	-	✓	✓ˣ
[72]	-	✓	-	-	✓	-	-	✓	-	✓	✓ˣ
[73]*	-	✓	-	✓	-	-	✓	-	-	✓	✓ˣ
[74]*	-	✓	-	✓	-	-	-	-	✓	✓	×
[75]*	-	✓	-	✓	-	-	-	✓	-	✓	✓ˣ
[76]*	-	✓	-	-	✓	-	-	✓	-	✓	×
[77]*	-	✓	-	✓	-	-	-	✓	-	✓	×
[78]*	-	✓	✓	-	-	-	✓	-	-	✓	×
[79]*	-	✓	-	✓	-	-	-	-	✓	✓	✓ˣ
[80]*	-	✓	-	✓	-	-	✓	-	-	✓	×
[81]*	-	✓	✓	-	-	-	-	-	✓	✓	×
[82]*	-	✓	-	-	-	✓	✓	-	-	✓	×

SG: Signature. SP: Specification. AN: Anomaly. HD: Hybrid
* Taken from our joint literature review paper [55] and not detailed in this thesis. ✓ˣ Partially

Their proposal demonstrates effective and efficient results in terms of detection rate, power consumption and memory usage. However, it has some false positive alarms [69]. Also, they consider a static-nodes-based environment, which is not common in IoT networks.

The authors in [65] design an IDS that can derive knowledge from a network and attack features to detect intrusions. The IDS can configure dynamically to provide proper detection mechanisms for the monitored network. In order not to affect the IoT nodes' performance, the detection, analysis and reporting modules are hosted in a separate (external) device placed near IoT devices. The detection process is based on signature, rules and anomaly detection. They involve different IoT network constraints such as heterogeneity, mobility, dynamicity and scalability. Even though Kalis, their proposed system, can detect routing and DoS attacks effectively, a single point of failure is still a concern.

This work [59] proposes an adaptive NIDS with a set of configured rules deployed on watchdog nodes to monitor the traffic pattern of neighbour nodes. The monitoring node runs in promiscuous mode to sniff packets sent from one node to another. In case packets match a set of a predefined set of rules, an alert is generated. An Event Management System (EMS) correlates alerts and decides on an intrusive node. Since the network is heterogeneous, they provide different rules for different monitor nodes. The IDS power consumption and the kind of attacks detected are not identified.

In another work [70], the intrusion detection modules are divided between the 6BR and distributed sensor nodes. The latter cooperate to detect any changes in their neighbourhood. Any malicious behaviour is sent to the border router for further analysis. The detection mechanism is based on the location and neighbour information. Also, signal strength is used to identify attacker nodes. During the deployment, the location information is set and monitoring nodes store information about their neighbour nodes. Any deviation is considered an indication of intrusion. Moreover, as the network is static, if the neighbours of a monitoring node are not in its transmission range that means an attack. They target the detection of wormhole attacks. In this attack, many DIO control messages are sent from one node to another. However, they deploy random placement with random detection modules. Even though they show how efficient their IDS is in terms of power consumption and memory usage, they focus only on one kind of attack.

The proposal in [71] is an extension of SVELTE [8] by incorporating dedicated detection modules to monitor the ETX metrics of each monitored node. These have already been defined by the network owner. Parent nodes always have lower ETX values and any changes are considered anomalies. They declare that this approach can effectively detect routing attacks and is efficient in memory usage and battery consumption.

An intrusion prevention system based on Complex Event Possessing (CEP) mechanism is proposed in [58]. CEP is a method of inferring a meaningful conclusion from a data stream by analysing its patterns. It aggregates different information to derive cause-and-effect relationships in real-time. The mechanism generates CEP rules that are used to detect DDoS attacks, namely SYN flooding, ICMP flooding, UDP flooding, and port scanning. They adopt a centralised architecture where the IDS resides at the network edge. To generate traffic, they run the IDS on a Raspberry Pi node with five desktop computers. Even though the proposed technique could effectively analyse a large stream of data, it has not been evaluated on low-resourced devices.

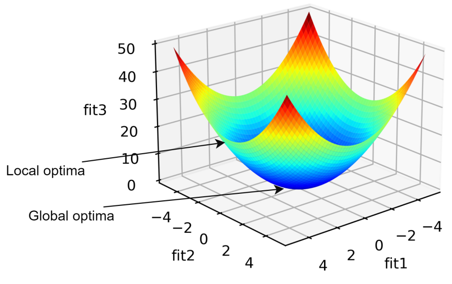
The work presented in [72] exploits the distributed passive monitored approach to detect anomalies in RPL networks. The monitoring nodes are dedicated to intrusion detection to preserve nodes' resources. They monitor the network for any topology attacks and inconsistencies. However, they do not consider the optimal placement of the passive monitors as well as the minimum required number to collect and make a decision. They require only that every node be within range of one monitoring node. Here, we investigate imposing requirements for higher levels of resilience (i.e. requiring each node to be within the range of $k > 1$ monitoring nodes).

2.4 IDS configurations optimisation

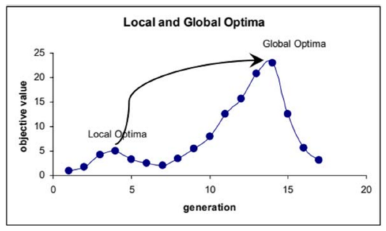
In large networks such as the Internet of Things, the configuration of an Intrusion Detection System (IDS), including sensors' placement, plays an important role in detecting unauthorised activities [83]. The resource-constrained nature of many devices and nodes restricts what tasks those nodes can realistically expect to perform. There may be a great many choices as to what detection functionality is allocated and where.

Placement problems of intrusion detection nodes have been intensively researched in the context of traditional networks. The problem is known to be computationally intractable and defining the best placement of a node is NP-complete [17]. Furthermore, finding the optimal configuration makes it even harder. There are several proposals for achieving the optimal IDS placement (and configuration to some extent) in WSNs (e.g.[17] and [84]). However, to the best of our knowledge, there is no such work in the field of RPL-based IoT.

There are different optimisation approaches in the literature. A major one is a branch of evolutionary algorithms (EAs) known as the Genetic Algorithm (GA). Next, we give a detailed overview of this algorithm followed by some related work in the context of IDS configurations optimisation.



(A)



(B)

Figure 2.16: *Local and global optima. (A) shows an example of a minimisation problem with multiple objectives; (B) shows an example of a maximisation problem with one objective).*

2.4.1 Genetic Algorithm overview

The Genetic Algorithm (GA) is a stochastic-based and meta-heuristic optimisation approach inspired by the natural selection (biological evolution) concept. The GA is a one type of the evolutionary algorithms (EAs) which is based on the principle of "survival of the fittest" that was first laid down by Charles Darwin. Moving the idea to solving real-world optimisation problems was first introduced by John Holland in his book "Adaptation in Nature and Artificial System" in 1975 [85]. This laid the foundation for later developments. However, the technique was not popular due to the lack of computation capabilities at that time. Nowadays, the algorithm is extensively used to solve a wide range of complex optimisation problems [86]. GAs are robust optimisation approaches that are different from other traditional optimisation methods in several fundamental ways [87, 88]:

- GAs search from a population of points, not a single point.
- GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge.
- GAs use probabilistic transition rules, not deterministic rules.
- GAs may converge to an optimal solution regardless of the chosen initial population. Conventional approaches, such as (gradient-based) hill-climbing, often get stuck in local

optima.

- GAs is a high-level problem-independent algorithmic (i.e., metaheuristic).
- GAs involve effective operators to avoid being stuck in suboptimal or local optimal solutions (see Figure 2.16).

Global optima are not guaranteed, but GAs have been found to give excellent results over a great range of optimisation problems. We opted here for a GA, and not any other meta-heuristic approaches, for the following reasons:

- It has flexible representations.
- It is easy to use (prototype) and implement.
- It is able to handle multiple criteria and able to handle more/fewer criteria as appropriate.
- It can facilitate the use of different multi-objective approaches (e.g., weighted sum or Pareto Front based approaches).
- It is computationally efficient [89].
- We are concerned with placement and configuration problems and has been considerable success in using GA from the direct literature [90, 91] or other areas.

Here we adopt a GA. However, the results of our work have good chances of applying to any optimisation approach for our target problem. For example, in Chapter 5, although our NN-based function approximation model provides fitness evaluation assistance to a hybrid GA-NN approach, the created approximation model can also be used similarly by simulated annealing to make simulated annealing more efficient via a hybrid SA-NN approach.
The flowchart in Figure 2.17 and the Algorithm 8 demonstrate the whole GA process.

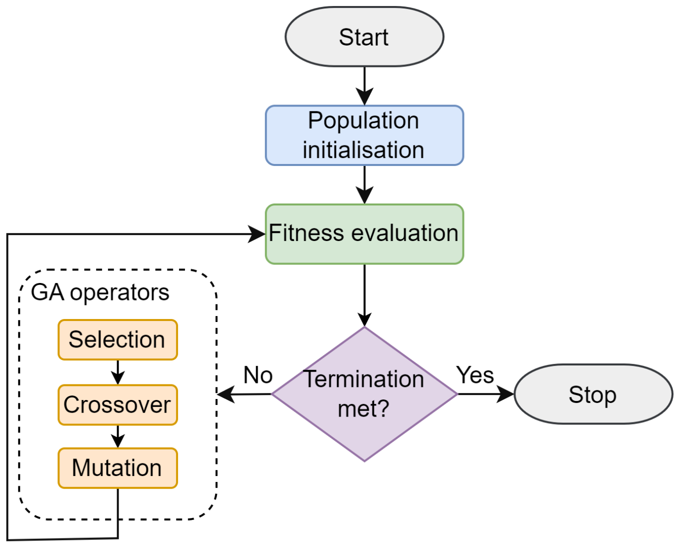


Figure 2.17: *Flow chart of a basic GA*

2.4.1.1 preliminary

In a genetic algorithm, each possible solution to a problem that needs to be solved is represented as a chromosome. A chromosome contains *genes* and *alleles*, which are the parameters of the problem at hand. The genes are the location of the solution's components while the alleles are the values taken by those components.

Encoding and decoding the chromosomes are crucial steps to mapping a genotype to a phenotype. A genotype (e.g., a string of bits) may be decoded into a phenotype (a potential solution from the solution space). The phenotype may then be evaluated. Some of the representation (encoding) methods are binary representation (a string of bits), real number representation, and integer representation. An example of a binary representation of a population is given in Figure 2.19. In this research, we adopt the binary (bitwise) representation for two main reasons:

- It is simple and easily manipulated.
- Many of our configuration (placement) issue is based on On/Off decisions.

Algorithm 8: Classical Genetic algorithm

```
Input: Population size, $Pop_{size}$;
       Generation size, $Gen_{size}$;
       Chromosome length, $Chrom_{size}$;
       Mutation rate, $Mut_{rate}$;
       Crossover rate, $Cros_{rate}$
Output: Global best solution, $Y_b$
1  Generate initial population of $Pop_{size}$ chromosomes $Y_i$ of length $chrom_{size}$,
   $(i = 1, 2, ..., Pop_{size})$;
2  Compute fitness of each solution $Y_j$;
3  Set counter $t = 0$;
4  Save them in the population $Pop_t$;
5  while $t < Gen_{size}$ do                          ▷ Or any other termination criterion
6      Select solutions from $Pop_t$ to $Pop_{t+1}$ ;        ▷ Based on the selection method
7      for $j = 1$ to $Pop_{size}/2$ do
8          Select two pairs of chromosomes from $Pop_{t+1}$;
9          Apply crossover by $Cros_{rate}$ ;              ▷ Based on the crossover method
10     for $j = 1$ to $Chrom_{size}$ do
11         Select one chromosome from $Pop_{t+1}$;
12         Apply mutation by $Mut_{rate}$;                 ▷ Based on the mutation method
13     Replace $Pop_t$ with $Pop_{t+1}$;
14     Compute fitness of each solution in $Pop_{t+1}$;
15     Increment $t$ by 1;
16 Return best solution $Y_b$;
```

2.4.1.2 Initialisation

The algorithm typically starts by initialising a set of candidate solutions randomly. (Less often, this set may be generated by some another technique.) Based on the specified number of population (Pop_{size}), this will construct a diverse population to form the first generation. The length of the chromosomes ($Chrom_{size}$) is based on the set of parameters that defines a candidate solution to the problem to be solved or optimised.

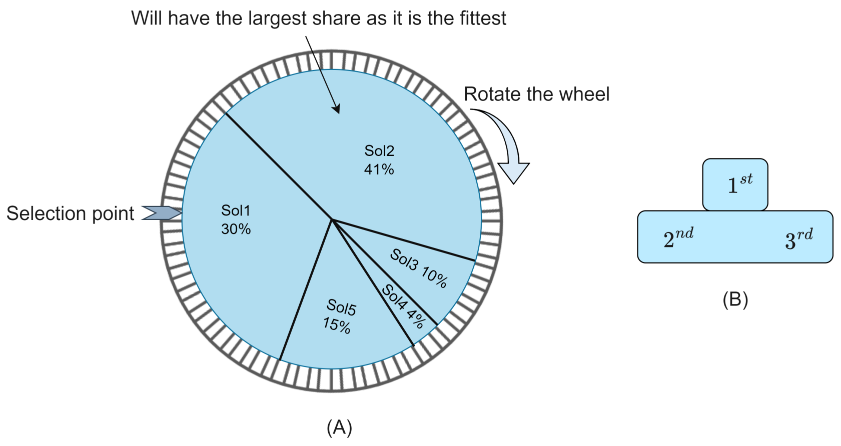


Figure 2.18: *Two common selection methods: (A) Roulette wheel selection (B) Tournament selection*

2.4.1.3 Evaluation

A fitness function needs to be defined to interpret the objectives being optimised. Each possible solution is evaluated against a fitness function. This will assign a fitness value to each solution that represents its goodness among other solutions in the population. Based on the optimisation requirements, there are some problem domains where the fitness function is either maximised (e.g. objective function), minimised (e.g. constraints) or both (multi-objective function).

2.4.1.4 Selection

Selection is one of the GAs' core operators (together with crossover and mutation) that helps the algorithm to find its way to optimal solutions. This is where the idea of "survival of the fittest" takes place. Two or more candidates (parents), with large fitness values, are selected to create a mating pool and pass on their genetics to the next generation. There are a couple of selection methods for the GAs. The most widely used approaches are the roulette wheel and tournament.

In the roulette wheel method, all individuals in the population are given a probability-of-selection value based on their fitness value. The larger the fitness value, the higher the probability-of-selection value they receive. For instance, in Figure 2.18 (A), there are 5

Algorithm 9: Algorithm of roulette wheel selection

```
Input: Population size, $Pop_{size}$
Output: Selected individuals
1 for $j = 1$ to $Pop_{size}$ do
2     sum_fit ← fitness
3     prob = (fitness/sum_fitness)
4     r = random (0,1)
5     if $prob > r$ then
6         select $j_{th}$ individual
7 create_offspring
```

Algorithm 10: Algorithm of tournament selection

```
Input: Population size, $Pop_{size}$
Output: Selected individuals
1 for $j = 1$ to $Pop_{size}$ do
2     selected = select $k$ individuals
3 for $i = 1$ in selected do
4     if $i_{th}$ individual > others then
5         select $i_{th}$ individual
6 create_offspring
```

candidate solutions that have been assigned with a probability value (converted here into percentage). The wheel will spin and the particular slot that it rests in will be chosen. Clearly, solution number 2 will be selected more frequently. This gives a chance (though it is low) for other chromosomes to be selected as well. The probability a specific individual will be selected is:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

where f_i is the fitness value of the ith individual and n is the number of individuals.

Tournament selection, on the other hand, is much simpler (see Figure 2.18 (B)). As the name indicates, a l (tournament size) number of solutions from the population enters

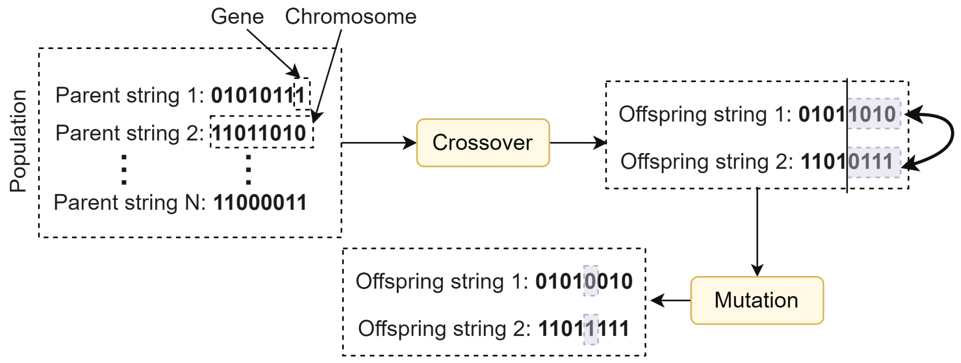

Gene
Chromosome
Population
Parent string 1: 01010111
Parent string 2: 11011010
Parent string N: 11000011
Crossover
Offspring string 1: 01011010
Offspring string 2: 11010111
Mutation
Offspring string 1: 01010010
Offspring string 2: 11011111

Figure 2.19: *The crossover and mutation operators*

a tournament game and the winner (i.e., the one with the highest fitness value) is selected, with or without replacement, to contribute to the mating pool. The common size of the tournament selection scheme is ≤ 4 (this is to avoid duplicate candidates [92]). Setting the size to 1 will lead to random selection while a large value may lead to a loss of diversity [93].

Whichever method is chosen, the algorithm will repeat the selection process until the number of the specified population is met.

2.4.1.5 Crossover

Sets of two parents from the mating pool are randomly nominated to undergo the crossover (also known as recombination) process by swapping their genes to generate new offspring. It is likely that the combined good parents will create better new individuals. The created offspring, of the exchanged parents, will form the population of the next generation. Common methods of conducting the crossover method are the simple crossover (single-point, two-points and multi-point cuts) and random crossover. Figure 2.19 shows an example of a simple crossover method, more specifically the single-point crossover.

2.4.1.6 Mutation

The main concept of mutation is to randomly alter a value (i.e., allele) of one or more genes. This is an important step so that the new generation might be different from their parents and create diversity within the population. For binary representation, the value of a gene will be mutated in a way that 0 becomes 1 and vice versa (as shown in Figure 2.19). Real numbers encoding mutation can be done by replacing the current allele with a random number from a range of min-max values of the problem. Mutation of an integer encoding could be performed by inversion, insertion or displacement mutation [86].

Choosing optimal parameters of the crossover and mutation rates is crucial for the GAs to converge to compelling results. For stability reasons, the mutation rate (Mut_{rate}) is preferred to be set to a very small value such as 0.01 [94]. Thereby, the algorithm will have more local search capability. A high mutation rate leads the search to be random. On the other hand, it is preferred to set the crossover rate ($Cros_{rate}$) to a high value to create more diversity in the population (i.e., more global search capability). For instance, the "0.03MR0.9CR" approach is a well-known predefined parameter setting for the crossover and mutation rates [94].

2.4.1.7 Termination

There are various termination criteria that will make the GA stop further searching. This is when any of the following is reached:

- Threshold or high fitness value.
- Maximum number of generations.
- High usage of computation resources.
- No further improvements after successive iterations (or specified delay).
- Manual interruption.
- Combinations of any of the above.

Unlike other search-based optimisation techniques such as brute force or random search, the GA is computationally cost-efficient [89].

2.4.2 Related work in IDS configuration optimisation

In this section, we present some relevant work on optimising the IDS configurations.

The authors in [17] propose a model to find the optimal placement and configuration of the distributed network-based intrusion detection system (NIDS) in WSNs. Their work is motivated by the infeasibility of deploying a single IDS in WSNs as the devices involved are constrained and there are a variety of IDS modules (tuning parameters). They used a heuristic-based search method to find the optimal solution. The model analyses and considers the trade-offs between the computation cost (e.g. resource consumption) and detection rate. However, it is only a mathematical model with no real attack evaluation. In addition, heterogeneity is not considered.

Another remarkable work is provided by [84] to help wireless sensor network operators to select the right detection configurations for their specific application. They propose a framework to obtain semi-automatic optimal IDS configurations in terms of detection accuracy and memory usage. They utilise the evolutionary algorithms (EAs), more specifically the eoEasyEA algorithm, to optimise the IDS configurations. They adopt the idea of majority voting where monitoring nodes overhear their neighbours and decide whether any of their neighbours are malicious or benign based on the number of packets received and forwarded. They consider one type of attack, selective forwarding.

Researchers in [95] propose algorithms to decide on either enabling or disabling the IDS in a specific node. Having all the nodes sniff all the packets passing through consumes significant resources. Thus, they propose minimising the number of activated IDSs that inspect packets while ensuring that every packet forwarded from a source node to a sink node is inspected at least once.

Multi-objective Genetic Algorithm (MOGA) has been presented by [90] to optimally find the IDS placement. The fitness function maximises the detection rate and minimises the number of nodes, false alarm rate, and communication costs. As they stated, different kinds of attack yield different optimal placements. The evaluation of any placement is either by information gain (e.g., location of the firewall and servers) or attack simulation. Therefore, they simulate a network and intrusive behaviour using NS2 where client and server nodes are grouped into different clusters. They aim to protect the more critical asserts; hence servers.

Their proposal, however, considers placement with no configurations.

Introducing a logic-based model to describe a network and its possible intrusions is proposed by [96]. Accordingly, it provides a suitable decision about the IDS sensors' placement and information needed to detect an attack. The properties of the network have been analysed and implemented in Prolog to infer the solutions. As they stated, NIDS comprises many sensors distributed on the network to monitor the traffic. Yet, to detect topology-dependent attacks, it is better to understand the underlying network topology. The model can automatically determine the location and/or the information needed by a NIDS sensor to detect the attack. They address the placement and configuration in general but not in constrained networks.

The authors in [97] propose using an attack graph to find the best location for placing nodes that hold an IDS. They adopt the idea of placing the IDS sensors only within the attack paths to critical mission assets. It might reduce the number of sensors deployed, and as a result, reduce the cost and overheads. The network configuration, including hosts, OS, applications and vulnerabilities, is scanned and then the predicted path is formed. They propose using a greedy algorithm to find the optimal selection of placements that covers all vulnerable connections for the attack graph. The more paths passing through a node, the more opportunity of placing the IDS at that node. Finally, they prioritise the alerts based on the distance of the attacker from the critical assets. Such that the closer to any valuable resources, the higher the priority. Nevertheless, the attack path approach may not address the optimal placements in a dynamic environment. Moreover, the detection technique and configuration are not discussed.

The authors in [67] define the characteristics of a network and nodes to determine the best deployment of IDS sensors in a network. The characteristics of the network are location type, load factor, risk profile and disruption cost. The characteristics of the sensors are interaction ability, efficiency and cost. Based on that, they specify the deployment value of a sensor. As they stated, the higher the deployment flexibility to minimise the potential risk, the lower the deployment cost. In other words, optimal IDS sensor placement aims to maximise the

Table 2.4: *Optimisation-based IDS proposals.*

Ref	IDS sensor placement		Targeted domain	
	Minimising placement	Resilient placement	Heterogeneous network	Constrained environment
[90]	✓	×	×	×
[17]	✓	×	×	✓
[98]	×	×	×	×
[84]	✓	×	×	✓
[99]	×	×	×	✓
[100]	×	×	✓	✓
[101]	×	×	✓	✓
[102]	✓	×	✓	✓
[95]	✓	×	✓	✓
[96]	×	×	×	×
[67]	×	×	×	×
[97]	✓	×	×	×
[103]	✓	×	×	✓

deployment value.

An essential criterion of an intrusion detection system (IDS) is that to be effective. It should identify a high proportion of threats while maintaining an acceptable false alarm rate. To achieve this goal, the IDS's model and algorithm must be calibrated or configured accordingly. The authors in [98] provide sveral variables that influence the ideal configuration. The first element is the IDS's performance as demonstrated by its ROC (receiver operating characteristics), which is a curve that combines detection accuracy and false alarm rate. The second element is the firm's cost of employing the IDS. The third component is the hackers' strategy. The possibility that a hacker will be detected influences their behaviour, which is dependent on the configuration of the IDS. The authors provide an optimisation model based on game theory that gives insights into optimum IDS configuration. They study the incorporation of strategic interactions between IDS, firm, and hackers in the development of optimal configuration and algorithm.

Another notable work is provided by [84]. The authors propose a framework to achieve semi-automatic optimal IDS configurations in terms of detection accuracy and memory usage. To perform the optimisation they use a population-based algorithm, an Evolutionary Algorithm (EA); more specifically, the eoEasyEA algorithm. They adopt the idea of majority voting where monitoring nodes overhear their neighbours and decide whether a neighbour is malicious or benign based on the number of packets received and forwarded. The aim is to optimise the following objectives: false positives, false negatives and memory usage. A weighted single-objective optimisation is used to solve the trade-offs between the objectives. The work is extended in [99] via the use of a multi-objective optimisation algorithm. They demonstrate the effectiveness of NSGA-II over SPEA2 in terms of speed of convergence and the number of evaluations. Their work does not incorporate minimising the number of deployed IDS sensors. Here, we investigate how to minimise the number of IDS sensors or monitoring nodes (generally leading to reduced costs of installing, managing, maintaining nodes and reducing communications traffic in collaborative approaches).

The authors of [100, 101] provide a genetic algorithm based optimisation technique for finding optimal cooperative IDS monitoring nodes in the constrained WSN. Different valuable properties such as energy consumption, event reporting delay, network coverage and data accuracy are studied to provide an optimal monitoring approach. They targeted a one-hop attack (i.e., a malicious node can only attack nodes within its radio range) of the following attacks: SYN/ICMP flooding, port scanning, and web exploits. These attacks could either exhaust resources, scan for open ports or disclose information. The authors consider a heterogeneous network where nodes have three levels of resources. These are, from less resourced to more powerful, Joined or orphan nodes, aggregators and leaders. As such, three types of intrusion detection engines are considered: lightweight (LWDS), medium (RE-DS), and heavy (HW-DS). They are employed by joined and orphan nodes, aggregators, and leaders respectively. They consider the network to be clustered and the cluster leader node is always powerful to operate heavy-duty intrusion detection. Networks can be in variant topologies such as mesh or star. Furthermore, this assumption of a clustered network with a more powerful cluster head cannot be always guaranteed in many LLN networks.

A proposal in [102] utilises anomaly-based distributed watchdogs to detect attacks targeting Bluetooth mesh networks. The detection of the Denial of Service attacks is based on observing each received packet by the watchdogs in N observation windows and calculating a set of features such as the average number and standard deviation of packets per source and destination address. The authors determine the optimal placement of the watchdogs based on some already recommended locations. However, there is a need for investigating all locations based on performing a series of placements experiments. This will ensure the generality of the evaluation where all locations are considered. Furthermore, they propose dedicated powerful devices for intrusion detection purposes; however, this is not always the case when it comes to constrained networks such as the LLNs where ordinary nodes may assign some monitoring and detection tasks.

The authors in [103] investigate the use of a neighbouring-based monitoring approach (i.e. watchdog) to detect blackhole attacks in Ad hoc On-Demand Distance Vector Routing (AODV) based networks. Nodes listen promiscuously to the transmissions of their neighbours and determine whether they correctly forward the packets they receive. If the monitored node does not transmit the received packets in a threshold time (t), the node is considered to be malicious, i.e. a blackhole node. To preserve the monitoring nodes' energy, they listen to their neighbours periodically and not continuously. The selection of the watchdogs takes place during the initialization phase based on the connectivity level. Nodes with more neighbouring nodes are assigned the duty of monitoring and intrusion detection. While this might be effective with slowly arranged networks, on other networks, where the density of information is different, this technique may fail. Monitoring nodes should be placed at locations that best achieve stated goals, such as a high detection rate, and not placed using properties such as the level of connectivity. (We appreciate the implied correlation but believe it is more beneficial to evaluate configurations based on direct measures of actual sought achievement rather than proxy measures.)

More limitations of the above works are presented in Table 2.4.

2.5 Function approximation

2.5.1 Overview

The expensive optimisation problem (EOP) is a term used to describe an issue where evaluating potential solutions is extremely expensive or perhaps unaffordable [2, 104]. This problem is common and applies to optimisation based approaches to the solution of many real-world applications. Evolutionary Algorithms (EAs) have been effectively utilised to optimise many hard real-world problems. (Later in this thesis we too will adopt a GA, a form of EA.) However, two main issues stated by [104] make deploying EAs to complex optimisation problems a difficult task. First, the processing time required for an optimisation algorithm to evaluate a huge number of fitness functions before converging to an adequate solution is high. Secondly, individual evaluations may be computationally expensive.
The overall cost of using EA for solving the EOP can be expressed as [2]:

$$total\_cost = O(N) \times O(C) = O(N \times C) \tag{2.7}$$

where the $O(N)$ is the time complexity of the algorithm to solve the problem (N is the number of evaluations needed to converge to a satisfactory solution) and $O(C)$ is the average cost of every expensive evaluation.

The aim is to solve EOP with a minimal cost. This is made by decreasing the $O(N)$ and $O(C)$ as much as possible while maintaining acceptable solutions. To reduce the expensive optimisation cost, there are three directions to follow (as illustrated in Figure 2.20): problem approximation and substitution (to reduce $O(C)$), algorithm design and enhancement (to reduce $O(N)$) and parallel and distributed computation (say this is PL, the Equation 2.7 becomes $O(N \times C)/PL$, where $PL \geq 1$).

Therefore, the function approximation, as one branch of problem approximation and substitution, seeks to define a function that closely matches ("approximates") an original function more cheaply. Approximating a function is needed for two main reasons:

- The underlying function is unknown and needs to be defined with the available data

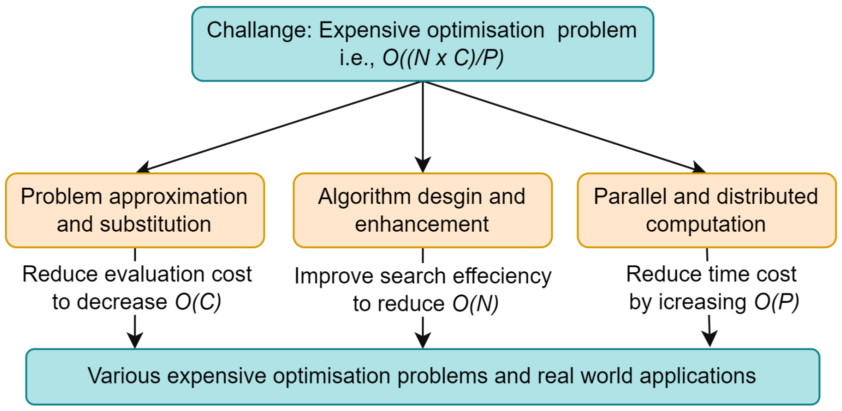

Challange: Expensive optimisation problem
i.e., *O((N x C)/P)*
Problem approximation and substitution
Algorithm desgin and enhancement
Parallel and distributed computation
Reduce evaluation cost to decrease *O(C)*
Improve search effeciency to reduce *O(N)*
Reduce time cost by icreasing *O(P)*
Various expensive optimisation problems and real world applications

Figure 2.20: *Three directions to reduce expensive optimisation costs [2].*

points.

- The computation of the original function is expensive.

In many real-world optimisation problems, there is no analytical fitness function for accurately evaluating the fitness of a candidate solution [3]. Instead, there is either a more accurate or less accurate fitness evaluation mechanism. Thus, this creates a trade-off between the computation cost and the fidelity, as shown in figure 2.21.

Even though approximation techniques can reduce the expensive evaluation cost, they may not evaluate the candidate solutions accurately (i.e., incur some estimation error). However, they can be useful in the evolutionary search (see Figure 2.22).

Many researchers have proposed using an approximation model (a surrogate model or meta-model) with evolutionary optimisation. The fitness approximation model has been used for fitness evaluation to reduce each fitness calculation or evaluation cost [105]. As a result, it may reduce the overall computation cost and time consumption. The fitness evaluation may involve an expensive simulation with a long iterative process. Many useful data points (or samples) are produced during the simulation process that can be used to train a surrogate model for predicting a fitness value for a candidate solution. An example of such an approach is given in Figure 2.23.

To obtain a global function approximation, the work in [104] asserts two measurements to be taken. The first one is that the original fitness function should be included within

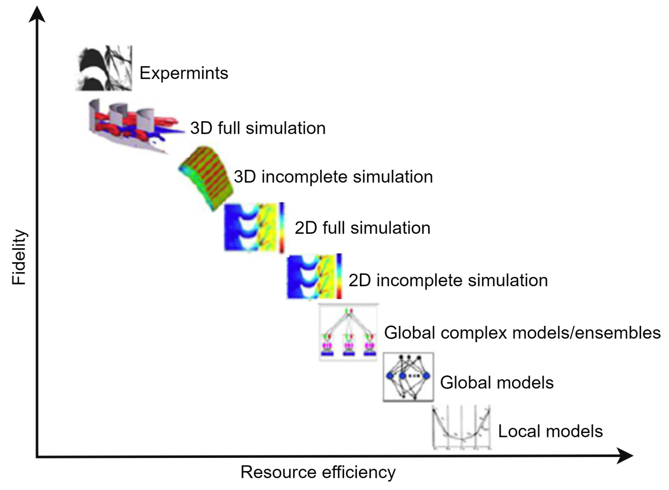


Figure 2.21: *An illustration of a trade-off between fidelity (approximation accuracy) and computational cost. Usually, high-fidelity fitness evaluations are more time-consuming. By contrast, low-fidelity fitness evaluations are often less time-consuming [3].*

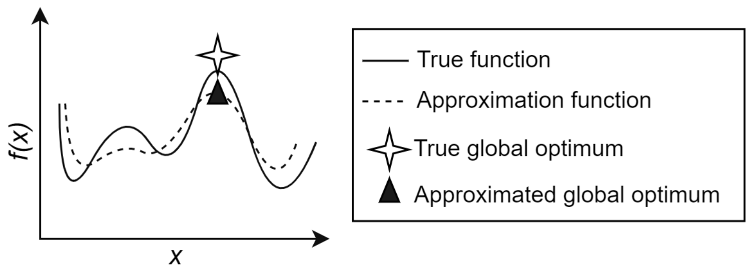


Figure 2.22: *An Example of a surrogate model that has a large approximation error but is adequately good for evolutionary search [4].*

the function approximation model, this is called model management or evolution control. Secondly, given a limited number of data, the performance of the approximation model should be enhanced as much as possible. A reduced quality could be due to poor choice of a model, training method, weights and error parameters. There are some obstacles that may affect the utilisation of an approximate model for fitness evaluation such as the high dimensionality of the optimisation space and the limited number of training samples.

The most widely used function approximation models are the polynomial model, the kriging model, the feed-forward neural network (multi-layer perceptron) and the support vector machine (SVM). In this PhD research, we target using NNs as a surrogate model for function

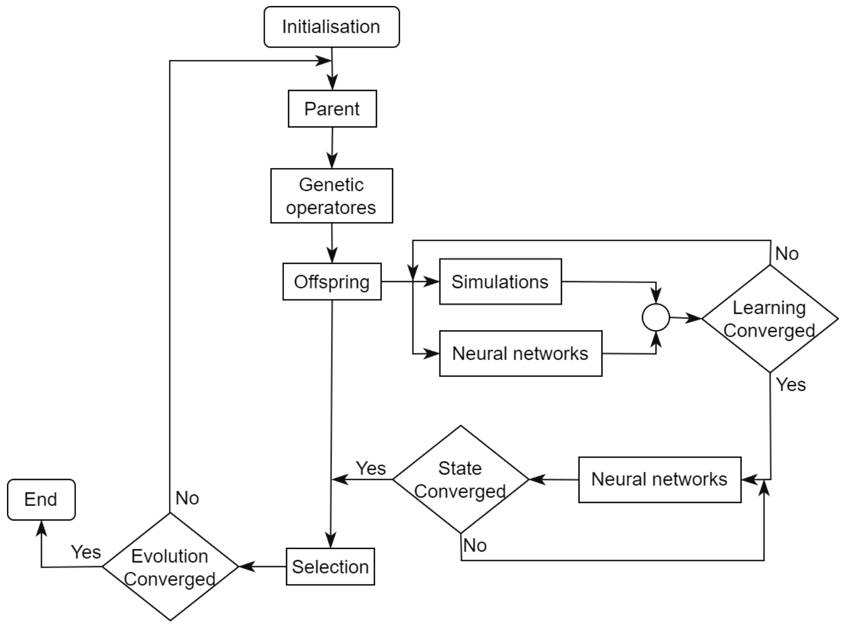


Figure 2.23: *An illustration of learning iterative fitness evolutionary process using neural networks for predicting converged fitness value [3].*

approximation, which is now addressed below

2.5.2 Fitness approximation via neural networks

Artificial Neural Networks (ANNs) have been proven to be an effective approach to approximate nearly any function to any degree of accuracy [19, 20, 104]. According to [106], Deep Neural Networks (DNNs), i.e. with several layers, are more efficient in function approximation than shallow neural networks (i.e. a single layer) even with a high number of neurons. In other words, extra hidden layers (depth) may exponentially reduce the number of neurons (size) for approximating a large class of functions.

Many proposals [106, 107] have adopted ANNs as a function approximator because of their powerful characteristics such as self-learning (adaptivity), nonlinearity, parallelism, fault tolerance and generalisation capability to map inputs to outputs [5]. When using ANNs for function approximation purposes, the model learns by training the algorithm on input-output samples to approximate the underlying rules connecting the inputs to the outputs. However,

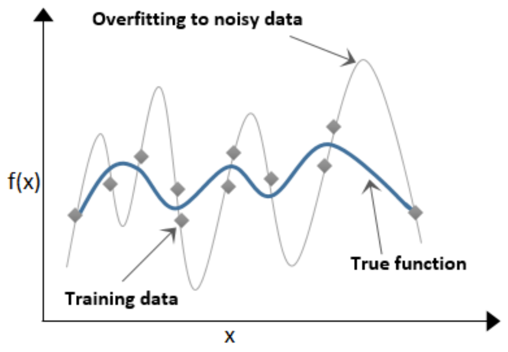


Figure 2.24: *An illustration of the Overfitting-to-noise issue.[5]*

there is a need for regulating the model complexity to avoid overfitting (as shown in figure 2.24).

2.5.2.1 Feedforward Neural Network

Feedforward Neural Networks (FNN) are a subset of Artificial Neural Networks (ANNs). An ANN network consists of *nodes or neurons* and *layers*. As shown in Figure 2.26, the information is passed from the input layer to the output layer passing via the hidden layer. Most of the calculations and computations happen in the hidden layer. In this layer, the nodes include the so-called activation function. This decides whether a neuron should be activated or not.

There are several activation functions in the literature, but here we will describe the most commonly used ones: *Rectified Linear Unit (ReLU)*, *Hyperbolic Tangent function (Tanh)* and *Sigmoid*. As illustrated in figure 2.25, the difference between these activation functions is in the output they produce or the conclusion they draw. The output of a node employing RelU as the activation function ranges between 0 to +infinity while the output of Tanh and Sigmoid ranges between -1 to 1 and 0 to 1, respectively. This function determines whether a node is "activated" ("fires") based on the weighted sum of its inputs.

A node i in layer l calculates its output S_j as:

$$S_j = \sigma_j\left(\sum_{i=1}^{k} w_{ij} \times l + b_j\right) \tag{2.8}$$

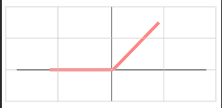
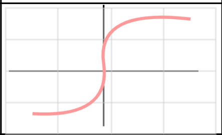
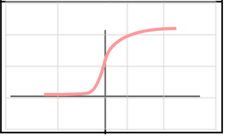
Name	Equation	Plot
ReLU	$f(x) = \begin{cases} 0 for\ x < 0 \\ 1 for\ x \geq 0 \end{cases}$	
Tanh	$f(x) = tanh(x) = \dfrac{2}{1+e^{-2x}} - 1$	
Sigmoid	$f(x) = \dfrac{1}{1+e^{-x}}$	

Figure 2.25: *Example of the activation functions*

where S_j is the output of node i, σ is the activation function, l is the jth input to the node i, w_{ij} is the connection weight between the node and input l, and b_j is the bias of the layer l. The NNs learn to map the input to the output via a comprehensive iterative process based on trial and error. In fact, there are two passes in the process of the neural network: *forward propagation* pass and *backpropagation* pass. The former is to fit the data to the NNs with some initial parameters (mostly randomly generated). Then based on the error rate (i.e. loss), between the predicted value (or classified class) and the actual value, the parameters are updated accordingly using the backpropagation method. This is used to adjust the weights, biases, learning rate (this controls how much these parameters should be changed with respect to the received error) and other parameters. This is called hyperparameter tuning or fine-tuning. It takes the gradient descent approach, using some optimisation algorithms such as Adam [108] and stochastic gradient descent (SGD)[109], to reduce the error to a minimal value. Each two-passes process (i.e., forward and backward) is called an epoch and many of them are performed until a satisfactory output (or a maximum number of specified epochs) is achieved. There are several loss functions (also known as cost functions) such as mean square error (MSE) and mean absolute error (MAE) for regression problems, and accuracy for classification problems.

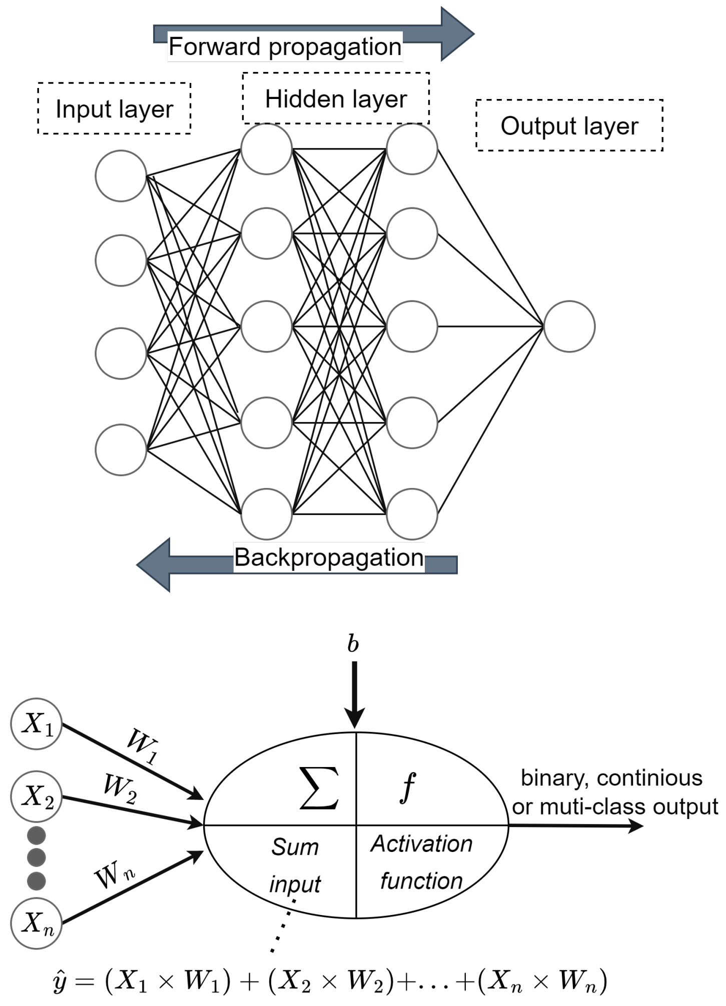


Figure 2.26: *Architecture of an ANN model*

The MSE measures the average of the squares of the errors or residuals. That is the average squared difference between the predicted value $\hat{y}_i$ and the actual value y_i.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where n is the number of observations.

The MAE measures the average of the errors or residuals. In other words, the average absolute difference between the predicted values $\hat{y}_i$ and the actual value y_i.

$$MAE = \frac{1}{n} \sum_{t=1}^{n} \mid y_i - \hat{y}_i \mid$$

In general, function approximation, or surrogate models, have been used to efficiently reduce the computation complexity for the EoP. However, it is also known that they may not provide accurate fitness evaluations. Therefore, there is a need to regulate the approach, mainly by increasing the number of training samples, to enhance the fidelity as much as possible while maintaining low computation complexity. In terms of the IDS configuration optimisation, we obtain the training samples via an expensive network simulation. This lies in the "2D full simulation" as shown in Figure 2.21.

2.6 Summary

IoT networks are adopted in many critical real-world applications, including health care, smart cities, smart homes, and industrial contexts. The nodes are resource-constrained and require efficient routing protocols to enable them to communicate their information efficiently. The RPL is proposed as the most commonly utilised routing protocol for the IoT. However, it is vulnerable to a variety of routing attacks. In this chapter, we detailed the RPL-based networks and their most common attacks. Furthermore, IDS proposals to mitigate such attacks for such a network were reviewed. We revealed the need for a specifically optimised IDS configuration (in terms of both functional and non-functional criteria) to satisfy the LLNs' requirements. IDS solutions for traditional networks cannot be simply utilised in the RPL networks due to the difference in the attack characteristics and available resources. We then provided a description of the evolutionary algorithm as a possible optimisation approach to optimise the IDS configurations along with some related work. Finally, a brief introduction to the field of fitness approximation and some applications were presented. Approximation models can be used in evolutionary algorithms with the aim to reduce the computation required.

Chapter 3

The Work of This Thesis

3.1 Brief motivation statement

Every detection of an attack improves security and any undetected attack may be costly. As demonstrated in Chapter 2, RPL networks are prone to many attacks. Intrusion detection systems are the security solution when other mechanisms fail to protect against attacks. However, they need to be configured appropriately to satisfy the resource constraints of IoT networks.

In Section 2.3.6 we reviewed many IDS proposals for IoT/RPL networks to establish the state-of-the-art. Usually, the presented IDS solutions focus on functional criteria such as enhancing the detection rate and reducing false alarm rates but omit non-functional measures such as the deployment cost.

There may be a huge number of possible configurations for an IDS for a system. Finding an optimum or excellent one may present significant computational difficulty. Section 2.4.2 showed that several attempts have been made to seek optimal IDS configurations for either conventional networks or WSNs but not for the RPL-based networks. To this end, an optimisation approach is explored in this thesis to optimise the IDS configurations for RPL-based IoT networks. Furthermore, although optimisation-based approaches have been highly successful across many domains, the computational costs may be prohibitive for certain applications. Making the search more efficient remains a challenge. In this thesis, we explore how leading concepts in function approximation (using a neural network) and model reuse (transfer learning) can be beneficially exploited. Thus, we bring leading contemporary machine learning concepts to bear on our target problem.

3.2 Research originality

The principal research hypotheses of this Ph.D thesis are restated below:

Hypothesis 1: *Evolutionary algorithms can discover resource-efficient and detection-capable security configurations for intrusion detection systems that are suitable for RPL-based Internet of Things networks.*

Originality: We propose an extensible optimisation framework that incorporates a much wider set of constraints (functional and non-functional) than other IoT/RPL IDS researchers consider. This is an original contribution in itself, but the framework also serves to generate data as input for the following work on approximation and transfer learning.

Hypothesis 2: *Machine learning approaches can allow us to perform function approximation for the framework's fitness evaluation function and so greatly reduce the time and computation taken to produce near-optimal security configurations using such a framework.*

Hypothesis 3: *A transfer learning based deep neural networks approach can provide a highly efficient fitness approximation with acceptable fidelity for newly-presented RPL-based Internet of Things networks.*

Originality (Hypo 2+3): To the best of our knowledge, there is no use of fitness approximation in the context of IDS configurations optimisation for IoT/RPL networks in the literature. Its use here is therefore original in its own right and the refinement via transfer learning is consequently also an original contribution.

Chapter 4

IDS Configuration Optimisation using Evolutionary Algorithm

As stated in the literature review chapter, Chapter 2, RPL-enabled Low Power and Lossy Networks (LLNs) involve constrained, unattended, and globally identifiable devices which require more optimised IDS solutions than traditional networks. In this chapter, we explore the application of a Genetic Algorithm (GA), a meta-heuristic-based optimisation technique, to identify the best IDS placement and configurations for LLNs. The effort is to optimise IDS settings for vulnerable and constrained networks while taking into account important objectives. The detection rate, F1 score, coverage, feasibility cost, and deployment cost are the objectives that our strategy aims to balance and optimise. We develope a framework that incorporates these functional and non-functional requirements.

4.1 Background

The Routing Protocol for Low Power and Lossy Networks (RPL) is the standardised routing protocol for the Internet of Things (IoT)[110]. It is used for the IP-connected wireless sensors that enable IPv6 over Low Power Wireless Personal Area Networks (6LoPANs). Such networks are also known as Low Power and Lossy Networks (LLNs). For these constrained devices to share their data to the outside world via the IPv6 border router (6BR) while preserving their power, they communicate in a multi-hop way. RPL-based networks are susceptible to several internal threats [111]. The nodes in LLNs lack effective self-protection due to low capabilities and resources [60]. Once an adversary gains access to a node, they may manipulate the node's functionality and operation to compromise the performance or the optimality of a network's topology. Some of the widely discussed attacks on RPL networks are Rank-based attacks (e.g. Blackhole and Selective Forwarding) and Denial-of-Service (DoS) attacks (e.g. DIS Flooding).

The Intrusion Detection System (IDS) is an important countermeasure when other security solutions fail to protect against attacks. It consists of several sensors or monitoring nodes distributed across the network for monitoring and detection purposes. As described in Section 2.3.1, IDSs are commonly divided into Host-based IDS (HIDS) and Network-based IDS (NIDS). The NIDS has a (often large) number of sensors to monitor the network. Effective intrusion detection for a large Internet of Things (IoT) network requires multiple monitoring sensors [112]. Detecting internal threats such as topology or routing-based attacks requires the IDS sensors to be placed not only on the gateway but throughout the network. The placement of the IDS sensors plays an important role in the accurate and timely detection of unauthorized activities [67][83]. Moreover, a more detailed configuration of these monitoring nodes (or probes) enhances their detection of malicious incidents.

Deciding where to place and how to configure IDS probes on a large network is a demanding task. The optimisation of several criteria may be simultaneously sought, e.g., maximising the detection rate while minimising deployment cost. The presence of conflicting criteria increases the difficulty of obtaining effective and efficient IDS. Furthermore, when it comes to a heterogeneous network such as the IoT [113], hosting the monitoring and detection tasks on constrained nodes may either be impractical or at least affect their principal functional operation. Last, but not least, in a rule-based IDS where rules are dependent on threshold values, finding effective thresholds is a difficult task, especially in dynamic environments [114]. A lower threshold value may increase the false positives while a higher one may come at the cost of more false negatives. To this end, we investigate the use of a meta-heuristic-based optimisation technique to discover excellent IDS placements and configurations with respect to chosen evaluation criteria. Specifically, we investigate the use of the GA to search the space and find optimal IDS sensors placements as well as their configurations. To the best of our knowledge, this is the first attempt to optimise IDS configurations for emerging and constrained networks such as the LLNs while incorporating a wider set of constraints than considered so far.

4.2 Our contributions and organisation of the chapter

In this chapter, we propose a meta-heuristic method based on a Genetic Algorithm (GA) to investigate IDS probes placement and configurations. The contributions of this work are summarised as follows:

- We present a GA-based framework for optimising the IDS sensors' placement and configuration.
- We present an effective fitness function targeting several valuable objectives such as:
  - Maximising the detection rate and F_1 score.
  - Minimising the number of IDS sensor nodes.
  - Minimising the hosting of resource-heavy intrusion detection tasks on the most constrained nodes.
  - Maximising the number of nodes within range of a specified number k of monitoring nodes (a detection resiliency measure).
- We investigate the allocation of the detection rules as well as finding effective firing threshold values for those rules.
- We investigate probe placement and configuration using predefined numbers of monitoring nodes and also investigate the automatic determination of the number of monitoring nodes by the framework itself.

Table 4.1: *Related works*

Ref	Scalability	Adaptive detection threshold	Constrained environment	Resiliency	Rules allocations*	Optimising placement	Parametrisable placement**	Heterogeneity
[103]	×	×	✓	×	×	×	×	×
[72]	×	✓	✓	×	×	×	×	✓
[90]	✓	×	×	×	×	✓	×	×
[17]	✓	×	✓	×	✓†	✓	×	×
[98]	×	✓	×	×	×	×	×	×
[84]	✓	✓	✓	×	✓	✓	×	×
[99]	✓	✓	✓	×	✓	×	×	×
[100]	✓	✓	✓	×	✓	×	×	✓
[101]	✓	✓	✓	×	✓	×	×	✓
[102]	×	✓	✓	×	×	✓	×	×
This work	✓	✓	✓	✓	✓	✓	✓	✓

*Or rules selection. **In terms of the number of IDS probes. †They consider different detectors (detection modules) allocations.

- We compare the results with the Random Search technique and show the effectiveness of the GA.

The chapter is organised as follows. In Section 4.3 we discuss the related work emphasising monitoring approaches and placements and configurations of IDS sensors. Section 4.4 describes our threat model and attacks targeted in this work. An overview of the IDS model, including the adopted monitoring approach, detection method and decision technique, are detailed in Section 4.5. The proposed GA-based IDS configuration is described in Section 4.6 followed by the performance evaluation and results in Section 4.7. Discussions and some future insights are given in Section 4.9. Section 4.10 summarises the chapter.

4.3 Related works

There have been several attempts to optimise the IDS placements and configurations in the literature. Chapter 2 presents an overview of the literature that relates to the work presented in this chapter along with their limitations. A more detailed comparison of related works is provided in Table 4.1

4.4 Threat model

Low Power and lossy networks (LLNs) (as in RPL-based networks) are resource-constrained by nature and are susceptible to several attacks [111]. Adversaries aim to disrupt the network communication and topology. The targeted attacks discussed in this work are the Blackhole, Selective forwarding and DIS flooding attacks. They affect the network in different ways, as described next.

4.4.1 Blackhole attack

Here, the attack has two stages. First, the attacker node decreases its rank in order to attract more nodes and traffic towards it. Neighbour nodes of the attacker node will be misled by the advertised low rank. Thus, they choose this node as their parent node and send their data via it. Next, it drops all the received packets. Algorithm 11 illustrates the process. In our experiments, we recorded its behaviour and effect on the network. The most disruptive and affected network performance metric is throughput. (All packets routed via the malicious Blackhole node are dropped.) Therefore, we monitor the Packet Dropping Ratio (PDR) to detect this attack.

4.4.2 Selective forwarding attack

Here the malicious behaviour is similar to the blackhole attack. However, the adversary node drops only selected packets to avoid being detected. As can be seen from Fig. 4.1 and Algorithm 11, the attacker could choose the probability of dropping and forwarding (i.e., low/high selective forwarding attack). The selective forwarding (also known as grey hole attack) is hard to detect in the context of LLN networks as packet loss, due to erroneous data transmission or network congestion, is naturally incurred in these networks [115]. To detect this attack the Packet Dropping Ratio (PDR) is carefully monitored.

Algorithm 11: Blackhole and Selective forwarding attacks

```
1 Initialisation
2 A: Attacker node, N: Neighbour list $\subset$ legitimate LLN nodes
3 B: a neighboring node $\in$ N, P: Current packet
4 R: a lower more powerful rank,
5 Attack_type = {Black-hole, Selective_forwarding}
  Input: "A" receives DIOs from A.N and calculates Min(advertised_ranks)
  Output: "A" obtains a lower, malicious rank and multi-casts it with DIO to all
          nodes, $\forall$ node $\in$ A.N
  if (P is DIO) $\wedge$ (P.sender_id $\in$ A.N) then
      if P.sender_id $\in$ A.N $\wedge$ P.sender_id $\neq$ root_id then
          if DIO.rank $\leq$ A.malicious_rank then
              A.malicious_rank $\longleftarrow$ R
  if (A.received(DIS from B)) $\vee$ (A trickle_timer activated) then
      A.multicast((DIO with malicious_rank) to node $\forall$nodes $\in$ A.N)
      B.receive(DIO from A)
      if DIO.rank < B.rank then
          B nominate A as preferred_parent
          B unicast application packets to its preferred_parent, which is "A" now, in
            order to transfer it to the destination
      if Attack_type is Blackhole then
          A collects packets from B then drops all
      else if Attack_type is High_Selective_Forwarding then
          A collects packets from B and randomly drops $\approx 70\% - 90\%$ of the received
            packets and transfers others to next hop
      else if Attack_type is Low_Selective_Forwarding then
          A collects packets from B and randomly drops $\approx 20\% - 40\%$ of the received
            packets and transfers others to next hop
```

4.4.3 DIS flooding attack

The attacker node in this kind of attack overwhelms the network with many DIS control messages. This affects the neighbouring nodes of the malicious node in terms of power consumption, more collided packets, and delay in the end-to-end packet delivery. Every time neighbouring nodes receive these DIS packets, they need to respond with broadcasted DIO control messages. In the normal scenario, the transmission of DIS messages happens when a new node wants to join the DODAG topology. They send enough DIS control messages to discover the neighbourhood and find parent nodes to connect to. However, the compromised node sends these control messages continuously. Furthermore, the attacker node may adjust

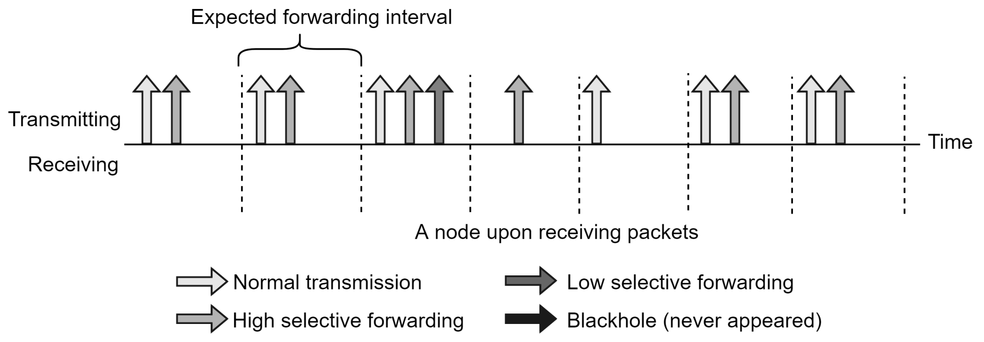

Expected forwarding interval
Transmitting
Receiving
Time
A node upon receiving packets
Normal transmission
High selective forwarding
Low selective forwarding
Blackhole (never appeared)

Figure 4.1: *Illustration of the blackhole and variants of selective forwarding attacks.*

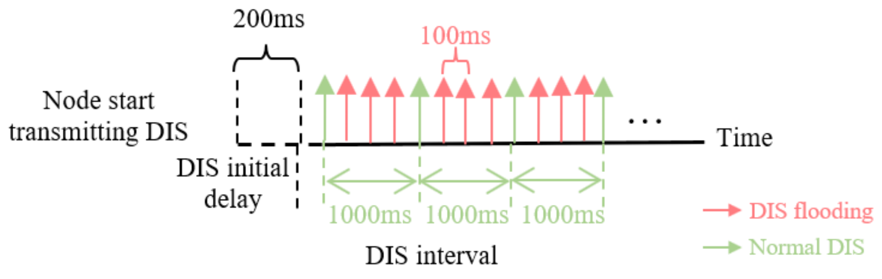


Figure 4.2: *The Difference in DIS intervals between normal and malicious scenario.*

the RPL trickle timer to increase the rate at which malicious DIS messages are sent and so cause more disruption to the network [75]. Fig. 4.2 illustrates such a technique and Algorithm 12 describes the process. Therefore, to detect the DIS flooding attack, we monitor the DIS Transmitting Rate (DTR).

4.5 Intrusion Detection System (IDS) model

In this section, we will describe the architecture of our proposed system. These are the monitoring technique, detection method and decision approach. These are the major components of our IDS model.

Algorithm 12: DIS Flooding attack

```
1 Initialisation
2 A: Attacker node, N: Neighbour list
3 B: a neighboring node $\in$ N,  P: Current_packet
4 Attack_type = {Multicast DIS Flooding}
5 Control_Packet = {DIO, DAO, DIS, DAO-Ack}
  Input: "A" multi-casts DIS to node(s), $\forall$ nodes $\in$ A.N
  Output: "B" multi-casts DIO message
  if P is DIS and A.Attack_type is Multicast DIS Flooding then
      A.Multicast(DIS, $\forall B \in$ A.N)
      set DIS interval to very low
      set time to infinity
      for $\forall B \in$ A.N do
          B.Multicast(DIO, $\forall$node $\in$ B.N)
```

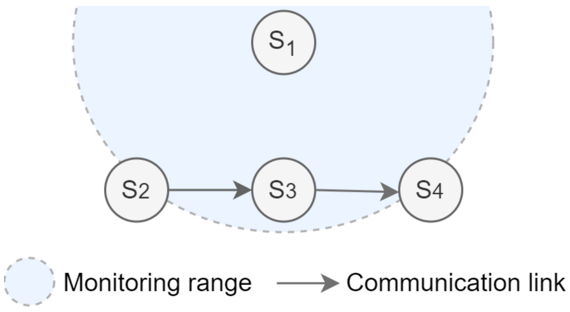


Figure 4.3: *The monitoring node, s_1, promiscuously overhears the communication of node s_3.*

4.5.1 Monitoring technique

Wireless sensors utilise their radio range for transmitting and receiving packets. These packets can be sniffed for either security or malicious purposes. We exploit this fact to investigate the placement of monitoring nodes or probes on the network. Each monitoring node (or watchdog [116]) monitors its neighbours within its radio range [117]. We are mainly adopting a Network-based Intrusion Detection System (NIDS) approach to monitor the network communication (i.e. packet traffic) and identify any abnormality. As can be seen from Fig. 4.3, a monitoring node s_1 can snoop on the in-bound and out-bound link from node s_3. The range of monitoring nodes may overlap. When calculating the functional evaluation metrics, detection rate and precision, we ensure that there is no "double counting" of malicious events.

4.5.2 Detection method

In this work, we adopt a rule-based intrusion detection approach to detect the aforementioned attacks. The rule-based detection approach is regarded as an effective and lightweight mechanism to detect RPL attacks [60]. A set of designed rules act as a baseline for normal activities and if any activity exceeds a developed threshold value (T), it will be reported as an intrusion. A rule for detecting a blackhole or selective forwarding attacks would require the monitoring nodes to generate an intrusion alert if:

EXIST a node x in any IDS sensor neighbourhood such that PDR(x) $\geq T$ *in t sec.*

The rule is to be varied to generate other rules to detect each of low/high selective forwarding attack; e.g., EXIST a node x in any IDS sensor neighbourhood such that PDR(x) $< T_{max}$ & PDR(x) $\geq T_{min}$ in t sec.

An alert will be generated by a probe for DIS flooding attacks if:

EXIST a node x in any IDS sensor neighbourhood such that DTR(x) $\geq$ *T in t sec.*

4.5.3 Decision approach

The monitoring nodes detect malicious behaviours locally and independently. This is for illustrative purposes; a variety of collaborative decision-making schemes can be incorporated into the framework. Furthermore, we are concerned solely with intrusion detection and not intrusion response.

4.6 Proposed GA-based IDS configuration

According to Goldberg [87], one of the pioneers of evolutionary algorithms, Genetic Algorithms (GAs) are robust search-based optimisation techniques. They are distinct from other traditional optimisation methods in several ways such as searching from a population rather than a single point and searching using stochastic or non-deterministic operators. It involves effective operators, such as crossover and mutation, to generate more diverse and fitter new individuals that can discover global optimum and avoid trapping in local optima. More benefits are given in Section 2.4.1.

Problem formulation: Placement problems of intrusion detection nodes have been researched in the context of traditional networks. The problem is known to be computationally intractable [17]. There are a great many choices as to what detection functionality is allocated and where. Hence, there are a great number of IDS configurations. The complexity has three sources: any node in the network is a potential placement of the probes and the resources available on each node affect the decision, each probe node could host any number of rules or even none depending on the resources available, and each rule has a (global) configurable threshold.

Encoding the problem: Candidate solutions have a binary encoding. The solution format is shown in Fig. 4.4. For each node the solution records which rules are active at that node: 1 indicates that the corresponding rule is activated and 0 means that it is idle. The solution also contains a binary representation of the (global) threshold value for each rule.

The number of the rules-related bits is based on the number of investigated rules. The bits at the end of each candidate configuration are for the threshold value to be applied to a corresponding rule. These are to be decoded into a non-negative integer value when implemented. For the node referencing, when all rules' indices of a node are 0s (i.e. Off), this indicates that there is no monitoring on this specific node. When the index of at least one rule is 1, then a monitoring-detection task is assigned to that node and is running the rule that is turned into ON. Therefore, the length of the chromosome is calculated based on the number of nodes, rules and threshold indices.

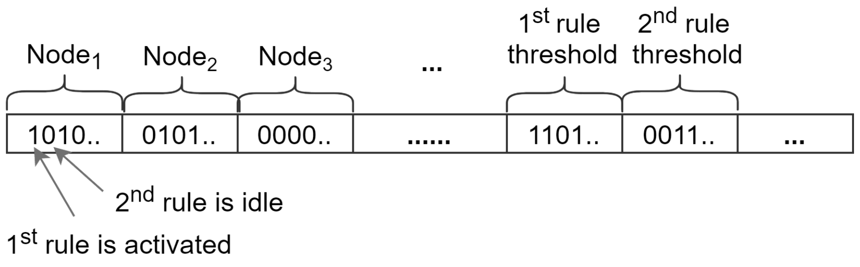


Figure 4.4: *The IDS candidate configuration representation.*

Initial population: The GA starts with a randomly generated initial population. This will allow the approach to start with a diverse set of feasible solutions. The length of the chromosome is calculated as follows. Let S=$\{s_1,s_2,s_3,\ldots,s_n\}$ be the set of the network nodes where n is the total number of nodes. s_j is the j^{th} node. Let R be the number of implemented rules. Each threshold value, T_j, is represented by a b-bit field, where b depends on the granularity needed to convert a binary set into a non-negative decimal number. The number of bits required for the threshold is then $Thr = b \times R$. Therefore, the chromosome length = (n $\times$ R) + Thr. The initial population is comprised of randomly generated bit-strings of this length.

Fitness function: To evaluate each possible solution, a fitness function is required. The fitness function evaluates each solution in terms of its performance regarding both functional criteria (here, detection rate and f1 score) and non-functional criteria (here, deployment cost, coverage and feasibility cost). In particular, we aim to optimise the following four objectives.

`Detection rate and F1-score` : Candidate configurations should provide a high capability of detecting attacks. Configuring the threshold value of the rules correctly, selecting the rules properly and placing the IDS sensors deliberately contribute to increasing the True Positives (TP) and reducing both False Negatives (FN) and False Positives (FP). We take into account the synchronization of the detected attacks where monitoring nodes work in a cooperative manner to eliminate the "double counting".

The detection rate (recall), is the fraction of actual attacks that give rise to alerts. It is a common and critical performance measure. However, high detection rates are practically traded off against precision (the fraction of alerts that are actual attacks). In our experiments, we trial two explicit targets: detection rate only, but also *record* the precision that results; and F_1 score, which gives an explicit equal weighting to detection rate and precision. In the latter, we explicitly target precision too. (F_1 *score* is usually defined in the machine learning community using the more general term *recall* in place of our more application-specific term *detection rate.* It is the harmonic mean of recall and precision.) The detection rate, precision and F_1 score are calculated as indicated by the following equations:

$$Detection\ rate(recall) = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F_1\ score = 2 \times \frac{Recall \times Precision}{Recall + Precision}$$

where TP indicates the correctly detected attacks, FN indicates the missed attacks, and FP indicates the incorrectly classified normal traffic as malicious.

Now, the first objective function will be formed as the following:

$$\begin{gathered} Objective1 = maximise\ fit1 = Detection\ rate \\ or \\ maximise\ fit1 = F1\ score \end{gathered} \tag{4.1}$$

`Coverage cost`: As stated earlier in Section 4.5.1, each monitoring node or probe monitors its immediate neighbourhood for any anomalous behaviour. As the targeted network is susceptible to many attacks and as these nodes (i.e. possible monitoring nodes) operate with low energy capabilities, they may turn off due to running out of energy or even being attacked themselves. We need to ensure the placement of the monitoring nodes is resilient. For this reason, we investigate requiring that each node be monitored by more than one monitoring node. We aim to incorporate a level of resilience into our monitoring operations and so we target k as the desired number of monitoring nodes within range of any node. Each node should be within the range of k monitoring nodes.

For each node s_j, let v_j be the number of monitoring nodes within the range of s_j. So, v_j is in the $set_j = \{0, 1, 2, ..., m\}$ where m is the number of neighbouring nodes. Let k be the target number of monitoring nodes within range, then $\forall\ s_j \in S$ we calculate the coverage of s_j as:

$$cov(s_j) = \begin{cases} k, & if\ v_j \geq k. \\ v_j, & otherwise. \end{cases}$$

There is no "reward" for additional coverage, i.e. for a node being within range of more than k monitoring nodes. Our requirement is viewed as a "satisficing" one. (We note also that it would be possible to adopt node-specific values for k, should there be compelling system-specific reasons to do so, but we do not do so here.)

We now build the second objective function:

$$Objective2 : maximise\ fit2 = \frac{1}{n \times k} \sum_{s_j \in S} cov(s_j) \tag{4.2}$$

`Feasibility cost`: to depict the real-world scenario of IoT networks, we consider a network of heterogeneous nodes. All the nodes are static in nature. These nodes differ in their capabilities; mainly in energy. As adopted by other researches (e.g., see[118]), we study three types of nodes: low-level energy (LE), medium-level energy (ME) and high-level energy (HE). Nodes at the high level are equipped with sufficient energy to hold more rules. Nodes at the low level cannot hold a probe. Nodes at the moderate level can hold a probe but with only a subset of the rules enabled (e.g., half of the ruleset R) and nodes at the high level can hold all rules. For practical purposes, nodes may be overloaded. Thus, we allow tradeoffs to be made with the understanding that if the loading on a node is 'infeasible' there may be significant effects on other aspects of that node's operation.

Let r_j be the number of rules that s_j is hosting and m_{rj} be the maximum number of rules that s_j can host; then, $\forall\ s_j \in S$ the feasibility cost is calculated as:

$$fCost(s_j) = \begin{cases} 0, & if\ r_j \leq m_{rj}. \\ (r_j - m_{rj}), & otherwise. \end{cases}$$

Where R is set to the total number of rules introduced to the system, the third objective function is defined as the following:

$$Objective3 : minimise\ fit3 = \frac{\sum_{s_j \in S} fCost(s_j)}{\sum_{s_j \in S} |R - m_{rj}|} \tag{4.3}$$

`Deployment cost`: There is always a cost for introducing an IDS sensor to a network, e.g., for installing, configuring or maintaining the sensor [67]. We assume here that all sensors incur the same deployment cost. Each node in the network is a potential placement of the IDS probes. We need to find the minimum number of probes placement and still fulfil k coverage. The next binary variables define the cost of hosting a probe on a node s_j:

$$depCost(s_j) = \begin{cases} 1, & if\ s_j\ is\ selected\ as\ a\ probe\ placement. \\ 0, & otherwise. \end{cases}$$

$$minimise \sum_{s_j \in S} depCost(s_j)$$

subject to:

$$v_j \geq k \tag{4.4}$$

The constraint (4.4) ensures that each node s_j is covered by at least k monitoring nodes. Therefore, equation (4.5) defines the fourth objective as:

$$Objective4 : minimise\ fit4 = \frac{1}{n} \sum_{s_j \in S} depCost(s_j) \tag{4.5}$$

4.6.1 Single-Objective Optimisation (SOO)

We next form our proposed fitness function. The above objectives may conflict with each other. Sometimes one needs to be maximised and others need to be minimised. Our fitness function works in a way that accepts the trade-offs by using a “weighted sum” method [119]. It is a well-known approach for converting a Multi-Objective Optimisation (MOO) function into a single scalar objective function. Furthermore, by varying the weight values, we illustrate

```
Algorithm 13: The implementation of GA
  Input: $Pop_{size}$, $Gen_{size}$, $Cross_{rate}$, $Mut_{rate}$, $Chrom_{node}$, $Chrom_{thre}$, $N_{nodes}$, $R_{rules}$,
         $T_{thresh}$
  Output: Best individuals
  for ($x$=1 to (($N_{nodes}$ × $R_{rules}$) + $T_{thresh}$) do
      $Pop_0 \leftarrow$ generate_initial_population()
  evaluate_initial_population($Pop_0$) (4.6)
  initialise_generation_counter(): $g \leftarrow 0$
  while ($g < Gen_{size}$) do
      select_from_population($Pop_0$): $Pop_{g+1} \leftarrow Pop_0$
      for $i = 1$ to ($Pop_{size}/2$) do
          select_two_parents()
          r $\leftarrow$ random_number_generator()
          if $r < Cross_{rate}$ then
              //$Chrom_{node}$ is the rules allocation part
              rc $\leftarrow$ random(1,$Chrom_{node}$)
              crossover($P_1, P_2$) on rc
              //$Chrom_{thre}$ is the threshold part
              rc $\leftarrow$ random($Chrom_{node}$,$Chrom_{thre}$)
              crossover($P_1, P_2$) on rc
          if $r < Mut_{rate}$ then
              rm $\leftarrow$ random(1,$Chrom_{node}$)
              mutate($P_1 and P_2$) at rm
              rm $\leftarrow$ random($Chrom_{node}$,$Chrom_{thre}$)
              mutate($P_1 and P_2$) at rm
          add_to_population ($P_1 and P_2$)
      evaluate_new_population($Pop_{g+1}$) (4.6)
      increment_generation_counter(): g $\leftarrow g + 1$
  output_best_individual()
```

the importance of each objective. As such, we build the fitness function as follows:

$$maximise \rightarrow fitness = (W_1 \times fit_1) + (W_2 \times fit_2) + (W_3 \times (1 - fit_3)) + (W_4 \times (1 - fit_4)) \quad (4.6)$$

where $W_1 + W_2 + W_3 + W_4 = 1$

$$i.e., fitness = \Big(W_1 \times detection\ rate\ or\ F_1\ score\Big) + \Big(W_2 \times \Big(\frac{1}{n \times k} \sum_{s_j \in S} cov(s_j)\Big)\Big) +$$

$$\Big(W_3 \times \Big(1 - \frac{\sum_{s_j \in S} fCost(s_j)}{\sum_{s_j \in S} |R - m_{rj}|}\Big)\Big) + \Big(W_4 \times \Big(1 - \frac{1}{n} \sum_{s_j \in S} depCost(s_j)\Big)\Big)$$

Table 4.2: *Genetic Algorithm parameters*

Parameter	Value	Parameter	Value
Population size	100	Population type	Bit strings
Number of Generation	200	Selection method	Tournament
Crossover method	Two-point	Crossover probability	0.9
Mutation method	Bitflip	Mutation probability	0.01

4.7 Performance evaluation

We study the placement and configuration of the IDS from different perspectives. We study the placement of the IDS sensors where there are a specific number of them and we need to find the optimal placement and configurations. This will eliminate the deployment cost as we do not need to search for the minimum deployed probes. Also, we investigate letting the framework decide on the number of probes. As stated earlier, we investigate the optimisation of the IDS in terms of a diverse and constrained network such as the LLN networks. Meaning, nodes incur variant levels of resources (i.e., a heterogeneous network).

4.7.1 Experiments settings

In this section, we will detail the settings used to conduct the experiments of the GA-based IDS framework and the network simulation to evaluate each possible IDS configuration.

4.7.1.1 GA Oriented Settings

We implement the Genetic Algorithm (GA) using its well-known genetic operators. these are selection, crossover and mutation. As illustrated in Algorithm 13, it starts with the random generation of possible solutions. A particular number of these (the population size) forms the initial generation of the population. These candidate solutions are evaluated using the fitness function described in Equation 4.6. Candidates are selected to go forward to the next generation using their fitness values, implementing the "survival of the fittest" aspect of the genetic algorithm. We adopt the tournament selection method. As advised in Section 2.4.1.4, it is more effective to set the tournament size to a relatively low value (but not 1). Thus, we set the tournament size to $l = 3$. This means 3 candidates are randomly selected from the

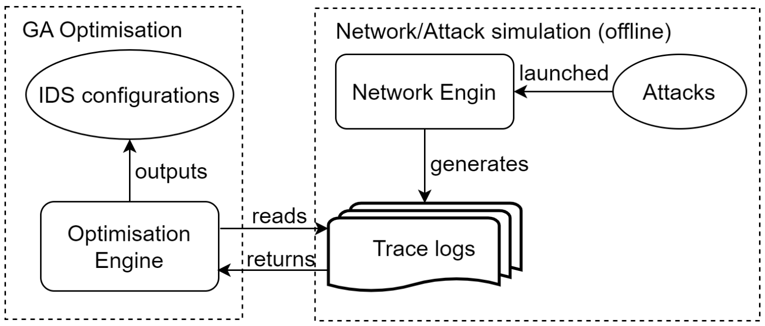


Figure 4.5: *The conceptual architecture of the framework.*

current population and the fittest of them forms the next offspring, going forward to the next generation of the population. Tournaments are held repeatedly (with replacement) until the next generation is complete.

The GA's crossover operation allows candidate solutions to "exchange DNA". Single-point crossover is applied to the rule allocations part of the genome and also to the threshold part. Thus, a point along the rule allocation part of the genome is selected for a pair of candidate solutions and the rule allocation elements to the right of that point are swapped between the genomes. Similarly with the threshold parts of the genome pair.

Our crossover points are chosen randomly with a high crossover probability to engage more parent candidate solutions to participate in the genetic recombination and form the new offspring.

Finally, since we adopt a bit-oriented representation for a solution (see Fig. 4.4), the GA's mutation operator is implemented as a bit-flip. Each bit in a solution can be flipped with a small probability. Table 4.2 illustrates the list of GA parameters used in our experiments.

The high-level overview of the framework is given in Fig. 4.5. The GA optimisation engine generates a set of feasible solutions to be evaluated. Each candidate solution is evaluated via a set of packet traces and log files generated from the network simulator. The optimisation engine computes the fitness function which involves the set of aforementioned objectives. A fitness value will be assigned to each individual which represents the goodness of the current IDS configuration in terms of both functional and non-functional criteria.

4.7.1.2 Network settings

We carried out the RPL network and attack simulations using NetSim v12.1 [120]. This is to construct the nodes' connections, protocols, communications, and attacks implementations. NetSim is a well-known network simulator of modern networks such as WSN and IoT. The RPL per RFC6550 [46] is supported in Netsim. We simulated a network of 64 nodes running the RPL as the routing protocol in the network layer. Nodes are heterogeneous in terms of energy. We simulated the heterogeneity as such 10% of the nodes were from the low-level energy nodes and 10% were from the medium-level energy nodes. Nodes are randomly assigned to either of these levels. The rest are assigned to the high-level energy nodes. The latter does not mean powerful nodes that can host a powerful IDS but rather powerful enough to hold the experimented 4 rules. (All nodes are assumed to be resource-constrained and hence adopting the rule-based IDS approach as it's known to be efficient (and effective) for the RPL-based networks [60].)

Table 4.3: *The network simulation parameters*

Parameter	**Value**	**Parameter**	**Value**
Simulator	NetSim v12.1	Node Type	IoT sensors
Number of Nodes	64 and 1 gateway	Channel Type	Wireless channel
Routing Protocol	RPL	Transport Protocol	UDP
Area	400m x 400m	Simulation Time	500s
Packet Generated at Source	500 application packets	Nodes initial energy	Varied (6480mJ,32400mJ,64800mJ)
Receiver sensitivity	-77 dBm	Distance between nodes$_{uniform}$	50m
DIS initial delay	200ms	DIS intervals$_{normal}$	1000ms
DIS intervals$_{attack}$	100ms	Traffic type	Sensing
Topology	Uniform/Random		

We also acknowledge that the distance between the nodes affects the performance, and we want to evaluate the robustness of the framework on variant networks in terms of topology. Thus, we investigated a random and uniform placement of the wireless sensor nodes. The distance between the nodes in the uniform scenarios is 50m. Precisely, the received sensitivity

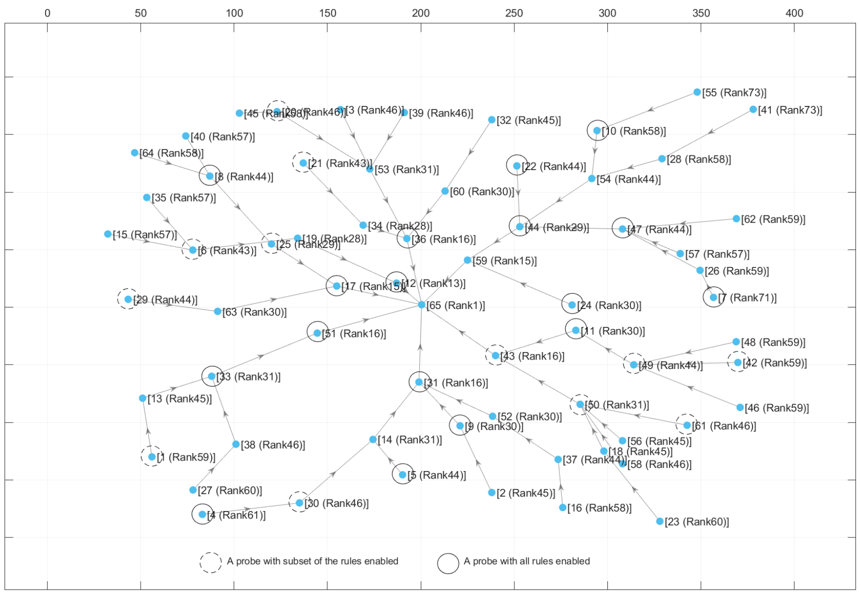


Figure 4.6: *The best found IDS sensor placement (random 64-node network with w_1=0.5,w_2=0.15, w_3=0.15 and w_4=0.2).*

is set to -77dBm which makes the radio range of the nodes be ≈ 50m. Sensors are configured to send data packets constantly to a destination server node at the rate of 1 per second. This will allow us to study the behaviour of forwarding packets and monitor the normal and anomalous behaviour properly. The packets are transmitted in a multi-hop way in order to reach the IPv6 Border Router (6BR). Table 4.3 lists the network simulation parameters.

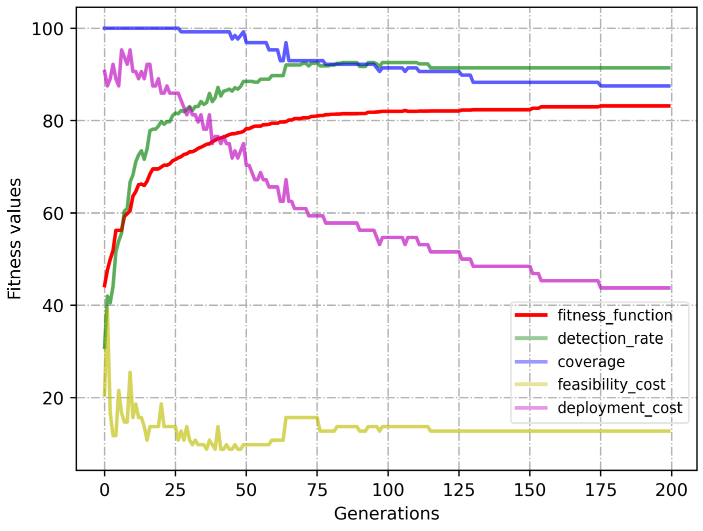
4.7.2 Results

To evaluate the proposed framework properly, we conducted an extensive set of experiments. The required computation to run these experiments is significant; thus we used our university High-Performance Computing (HPC) cluster. The execution time complexity for each single IDS configuration evaluation can be represented as $O(n \times np \times R)$; where it scales significantly with respect to the number of nodes n, the number of packets np and the number of rules R.

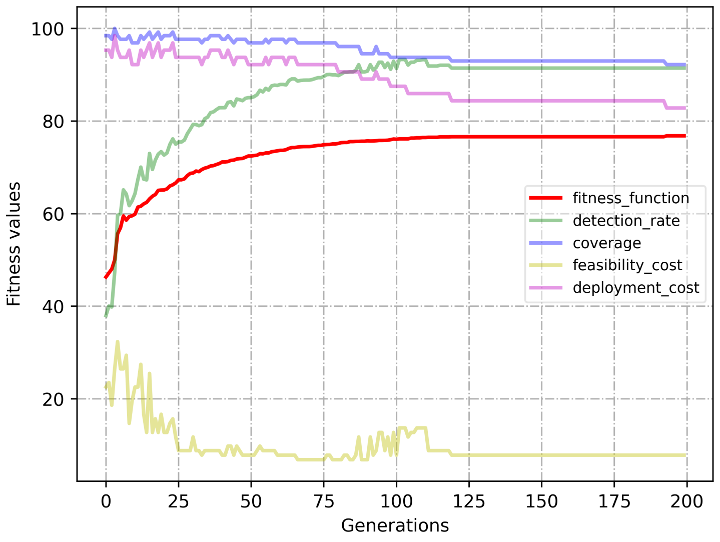
We have carried out extensive simulations, with each node in turn playing the role of attacker. This is because we aim to be resilient against any node's compromise. We run the attack with different attacking rates, blackhole attacks and low and high selective forwarding attacks. Hence, variant rules and in total four rules are investigated (as described in Section

4.5.2). Similarly, and as stated in Section 4.4, the transmission interval of the DIS control packets is varied between the normal nodes (i.e. new nodes that want to join the network) and DIS flooding-based attacker nodes. This is to study the ability of the framework to find optimal rule firing-threshold values as well as rules allocations. Thresholds are determined on a rule-type basis; there is a threshold for each rule type and all deployed instances of that rule type have the same threshold. This is for illustrative purposes only; node-specific rule-thresholds *could* be adopted.

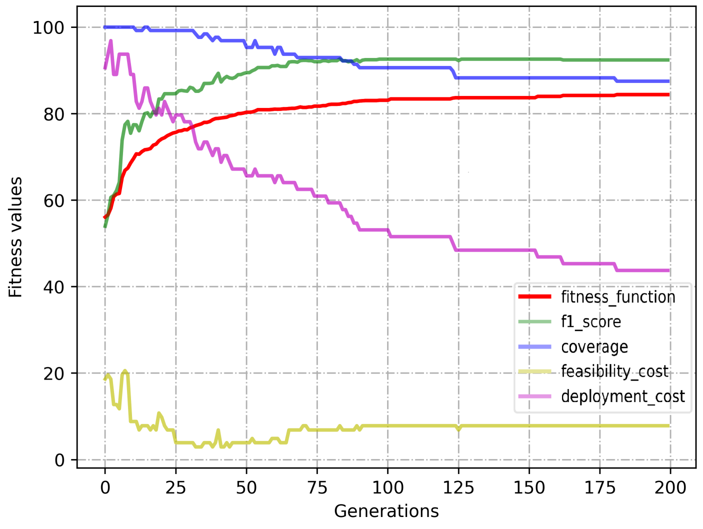
From Table 4.4, we can find that the threshold value is evolved and that the algorithm was able to converge to a high detection rate and f1 score; especially when they were more highly weighted. Our approach was able to discover high-performing threshold values for the aforementioned attacks. In Fig. 4.6, the proposed IDS sensors placement and rules allocation are presented with respect to the given weights for each objective. Rules allocation is dependent on the resources available on the node a probe is placed on. We set the $m_{rj} = 0, R/2 \ and \ R$ to the nodes in low-resourced, moderate-resourced and high-resourced levels respectively. To avoid overwhelming the network with monitoring nodes and still provide a plausible measure of resilience, our target monitoring coverage was set to $k = 2$, i.e. each monitored node should be within a range of two monitoring nodes.



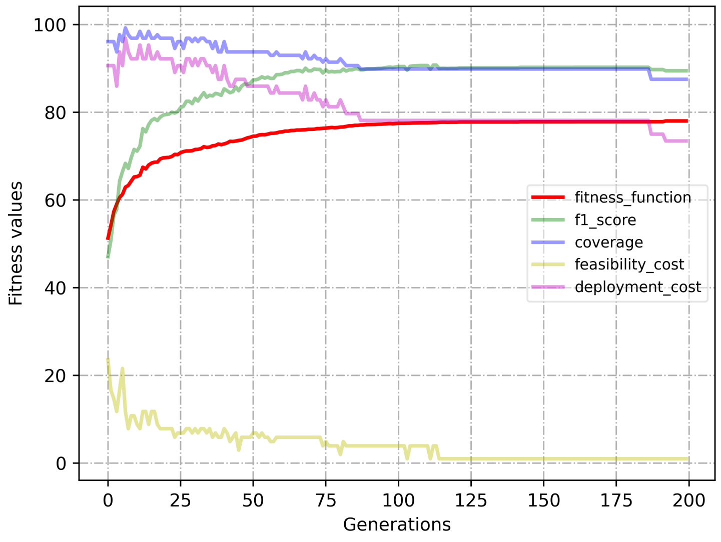
(a) *64-Node (Random)*



(b) *64-Node (Uniform)*



(c) *64-Node (Random)*



(d) *64-Node (Uniform)*

Figure 4.7: *The evaluation of the objectives and the fitness function over generations for different networks with w_1=0.5, w_2=0.15, w_3=0.15 and w_4=0.2 (Fit1 as detection rate (top two figures) and Fit1 as f1-score (bottom two figures)).*

We investigate different placement-configuration approaches. One approach is when we have a fixed number of probes, and we want to find the optimal placements and configurations. Another approach is when we do not restrict the number of monitoring nodes and let the framework decide on the number of them. We furthermore, consider different weights to illustrate the importance of each objective and how they can be varied to satisfy specific system needs.

From Table 4.4, we can see that when there is a parametrised number of probes we omit the deployment cost as the framework does not have control over that (i.e. to increase or decrease it). Hard-coding the number of monitoring nodes may fail to achieve appropriate coverage and detection (i.e. if the number is too low), or otherwise be wasteful (if the number is too high). Allowing the framework to choose the optimal number of them may be more efficient as it will take into consideration all objectives. However, if an administrator has a specific financial or operationally practical budget in mind, then fixing the number of probes may serve a useful purpose. This further illustrates the flexibility offered by our approach.

Table 4.4: *Various parameter and weight values and their achieved fitness values (%) using SOO*

					Detection rate as Fit_1								F_1 score as Fit_1							
	Weight values				Random				Uniform				Random				Uniform			
No.	W_1	W_2	W_3	W_4	Fit_1(Pr*)	Fit_2	Fit_3	Fit_4	Fit_1(Pr)	Fit_2	Fit_3	Fit_4	Fit_1(Dt,Pr)*	Fit_2	Fit_3	Fit_4	Fit_1(Dt,Pr)	Fit_2	Fit_3	Fit_4
	0.1	0.45	0.45	-	70.1(80.2)	60.9	0.0	15.6	34.0(76.8)	57.8	0.0	15.6	79.6(67.4,97.2)	60.9	0.0	15.6	49.4(33.2,96.1)	57.8	0.0	15.6
10	0.4	0.3	0.3	-	70.4(75.8)	60.9	1.0	15.6	36.1(78.0)	57.8	0.0	15.6	80.8(69.3,96.9)	60.9	0.0	15.6	51.2(35.5,91.7)	57.8	0.0	15.6
	0.8	0.1	0.1	-	74.6(83.4)	60.9	16.7	15.6	39.3(69.6)	57.8	9.8	15.6	81.8(71.3,95.8)	60.9	15.7	15.6	52.7(36.2,97.1)	57.8	0.0	15.6
	0.1	0.45	0.45	-	81.6(80.5)	88.3	0.0	46.9	71.9(82.8)	74.2	0.0	46.9	88.0(80.2,97.5)	88.9	0.0	46.9	80.6(68.6,97.5)	74.2	0.0	46.9
30	0.4	0.3	0.3	-	87.9(86.3)	88.3	3.9	46.9	73.8(71.4)	74.2	0.0	46.9	90.1(83.7,97.6)	88.9	2.0	46.9	81.1(69.4,97.5)	74.2	0.0	46.9
	0.8	0.1	0.1	-	95.4(81.5)	88.3	23.5	46.9	77.0(81.1)	74.2	11.7	46.9	95.3(93.3,97.5)	88.9	20.6	46.9	86.9(78.2,97.7)	74.2	11.8	46.9
	0.1	0.45	0.45	-	84.5(89.0)	100	1.0	93.8	88.2(72.7)	97.7	1.0	93.8	89.2(82.9,96.6)	100	1.0	93.8	92.2(88.1,96.7)	97.7	1.0	93.8
60	0.4	0.3	0.3	-	87.1(74.9)	100	2.9	93.8	89.4(81.5)	97.7	1.9	93.8	90.5(84.3,97.6)	100	1.9	93.8	93.4(89.6,97.6)	97.7	2.9	93.8
	0.8	0.1	0.1	-	95.4(75.3)	100	23.5	93.8	96.6(82.3)	97.7	24.5	93.8	95.3(93.3,97.5)	100	20.6	93.8	95.7(93.6,97.8)	97.7	13.7	93.8
-	0.1	0.3	0.3	0.3	72.3(80.0)	89.8	0.0	50.0	73.0(81.6)	81.3	0.0	60.9	84.6(74.8,97.4)	87.5	0.0	43.8	75.7(61.6,98.2)	77.3	0.0	53.1
-	0.3	0.25	0.25	0.2	85.5(82.2)	96.9	2.0	67.2	86.6(82.2)	92.2	1.0	82.8	87.7(79.5,97.7)	96.7	0.0	67.2	90.2(84.4,96.8)	89.4	0.0	78.1
-	0.5	0.15	0.15	0.2	91.4(81.2)	87.5	12.7	43.8	91.4(80.8)	92.2	7.8	82.8	92.4(87.7,97.4)	87.5	7.8	43.8	90.4(83.3,96.6)	87.5	1.0	73.4
-	0.7	0.1	0.1	0.1	94.6(78.9)	92.2	22.5	56.2	94.3(74.6)	93.7	18.6	85.9	92.5(88.0,97.4)	89.8	11.8	50.0	95.0(92.3,97.8)	92.2	13.7	82.8
-	0.9	0.04	0.03	0.03	96.5(89.0)	96.8	35.3	67.2	98.1(81.0)	96.7	53.9	92.3	96.1(94.9,97.2)	96.9	35.3	67.2	96.7(95.7,97.7)	96.7	24.5	90.6

* No. (Number of probes), Pr (Precision) and Dt (Detection rate)

A list of objectives and fitness evaluations of different networks with different weights is provided in Fig. 4.7.

We compared the results of the *Genetic Algorithm* with the *Random* search (RS). As can be seen from Fig. 4.15, the GA approach outperforms the RS approach. Exhaustive search, as one of the common search techniques, could be brought to bear; however, brute force is simply infeasible, especially when there are many parameters to tune [121]. Given enough time, RS explores all possible combinations; nevertheless, it could not achieve the high-performing configurations obtained by the GA in the given number of evaluations. Even for networks of moderate size, some heuristic approach must be used.

4.8 Interpreting IDS configuration samples

In this section, we consider 5 examples of high-performing solutions found in this work and look for points they share. These high-performing configurations are taken from 5 runs of the optimisation framework on the same network. Figure 4.8 indicates what each bit of each configuration represents. The number of times a specific node is enabled for probe placement is illustrated in Figures 4.9 and 4.14 (with $w_1 = 0.5$, $w_2 = 0.15$, $w_3 = 0.15$ and $w_4 = 0.2$). We can clearly see that there are nodes (e.g., 31, 33 and 43) in strategic locations that have been enabled for probe placement in all runs. On the other hand, there are only few nodes that are never chosen for probe placement (e.g., 2, 3 and 13). Figures 4.10, 4.11, 4.12 and 4.13 show the number of times a rule is selected on each node over the 5 different runs.

Most nodes have at least one rule enabled. Very few have none enabled. This might seem a little unusual but there are good reasons why this is so. For example, the requirement to have redundancy in monitoring inevitably leads to more nodes hosting probes. Additionally, we make an assumption that all nodes share the same wi-fi range. Deployments are a function of that range. If the wi-fi range were larger, then the patterns of rule distributions might change significantly. Even if significant patterns of deployment could be identified for this network with its specific parameter choices, it is far from clear that they would remain applicable when those parameters are changed. Also, if different nodes had different wi-fi ranges then this would add further difficulty to extracting high-performing and generalised deployment and configuration patterns.

It is clear from the results reported in this section that for a specific network there will be a large degree of commonality in the configurations obtained by repeated runs. However, determining a generalised characterisation of high-performing configurations would require analysis of very many runs over many different networks with varying parameters. Such analysis would typically require an ML-based approach to characterisation. Our considerations here would suggest that optimal configuration mining is potentially a fruitful task for the Explicable Machine Learning community. Indeed, we recommend it to that community.

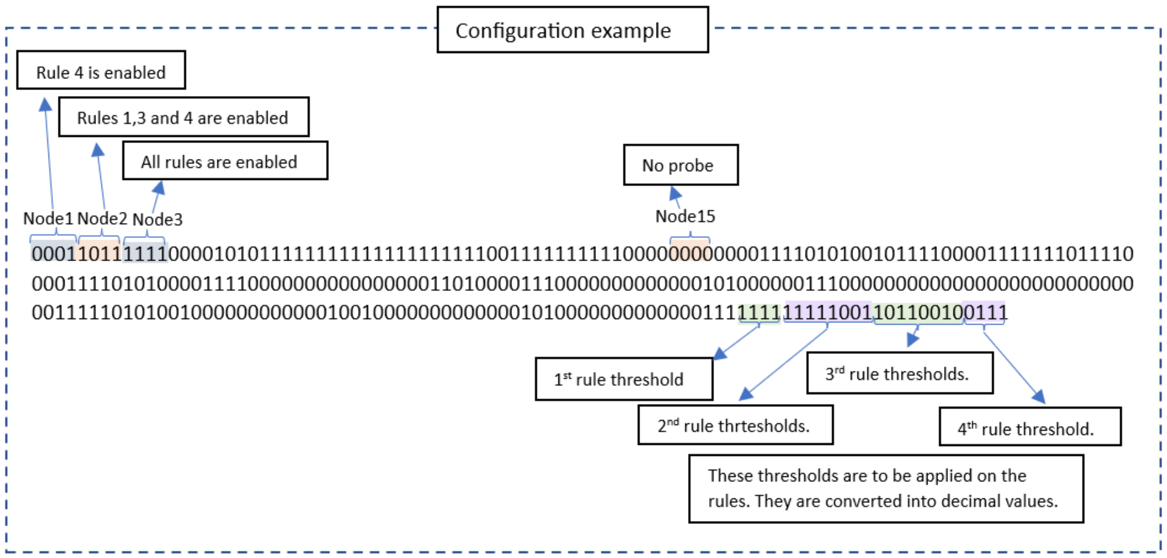


Figure 4.8: *An example of high-performing IDS configurations*

4.9 Discussion

Overall, when problems get sufficiently complex there is often no feasible alternative to adopting a heuristic of some form. In this work, we have shown that optimisation-based approaches can be adopted to advantage to address problems of considerable significance in modern-day networks. Were we to significantly extend the number of probes or the number of available rules, it seems unlikely that any other approach could identify high-performing deployments

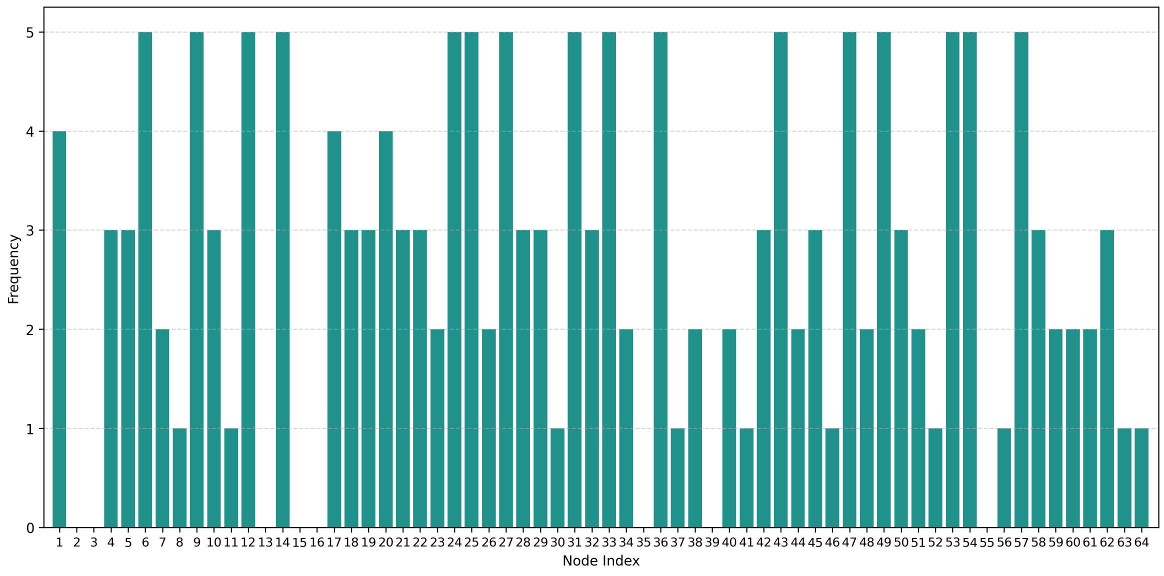


Figure 4.9: *The number of times a node is enabled for 5 different runs*

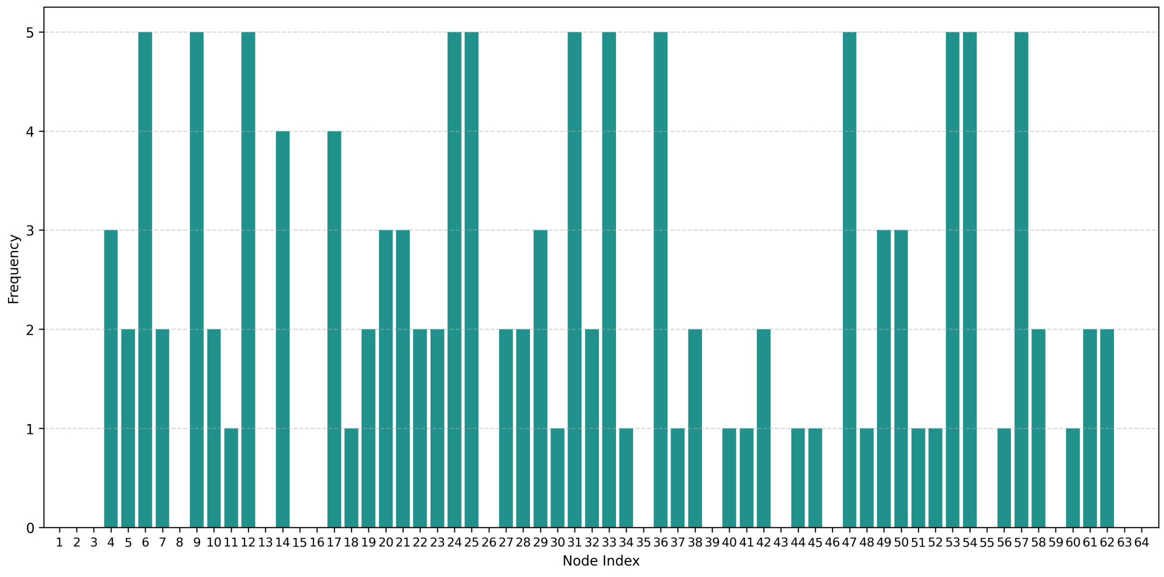


Figure 4.10: *The number of times rule 1 is selected on each node over 5 different runs*

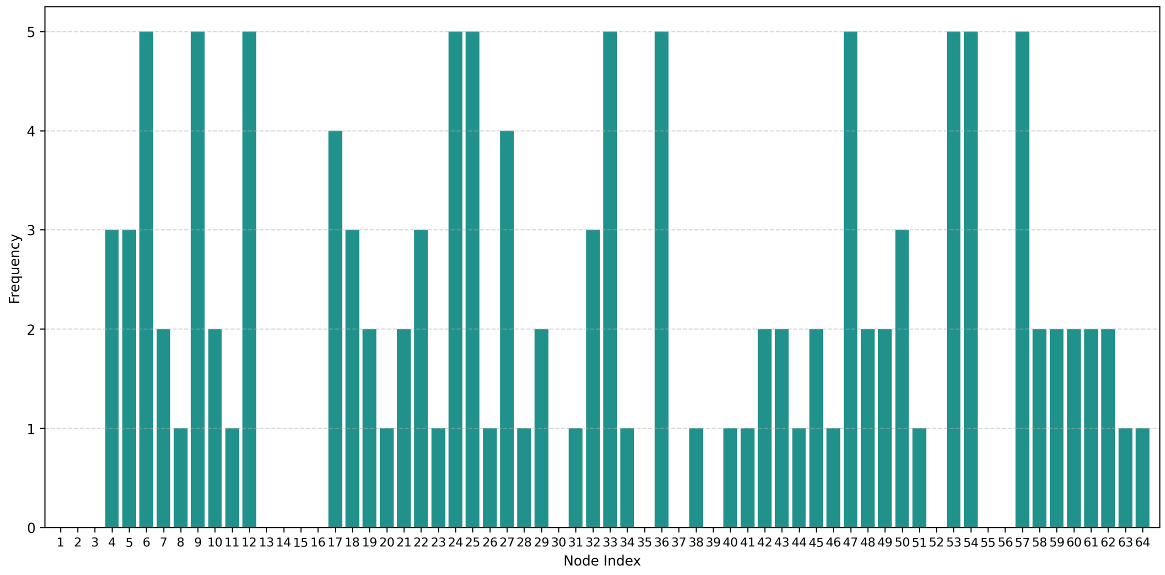


Figure 4.11: *The number of times rule 2 is selected on each node over 5 different runs*

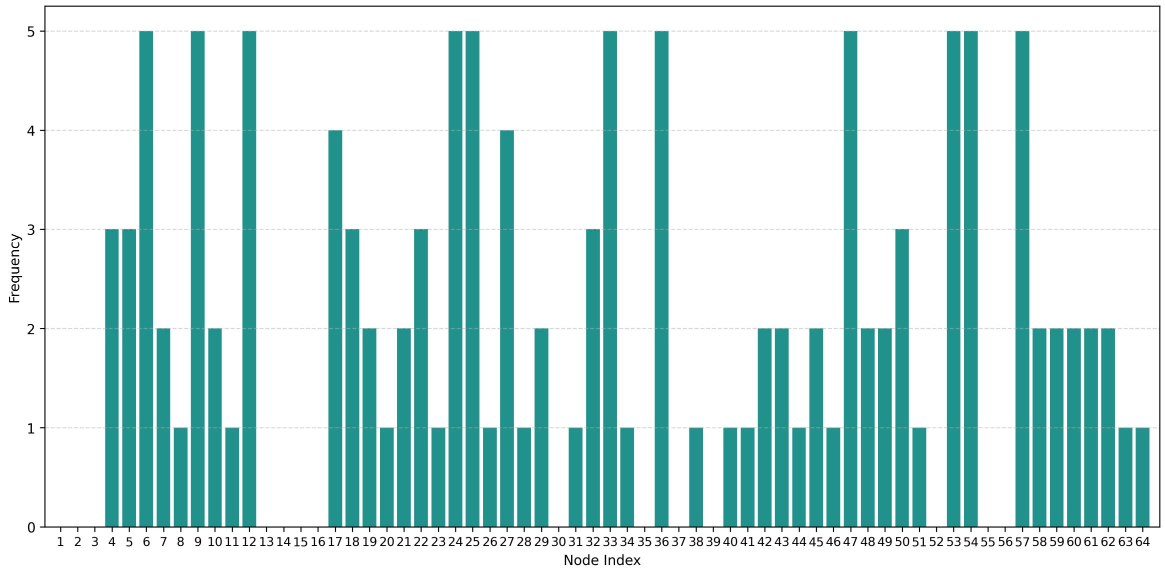


Figure 4.12: *The number of times rule 3 is selected on each node over 5 different runs*

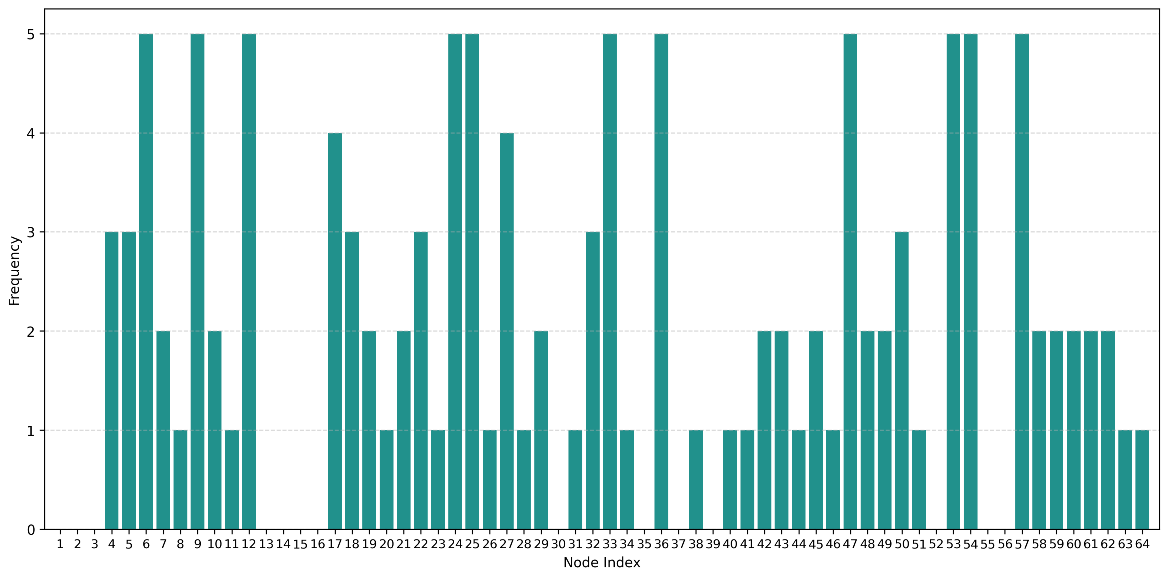


Figure 4.13: *The number of times rule 4 is selected on each node over 5 different runs*

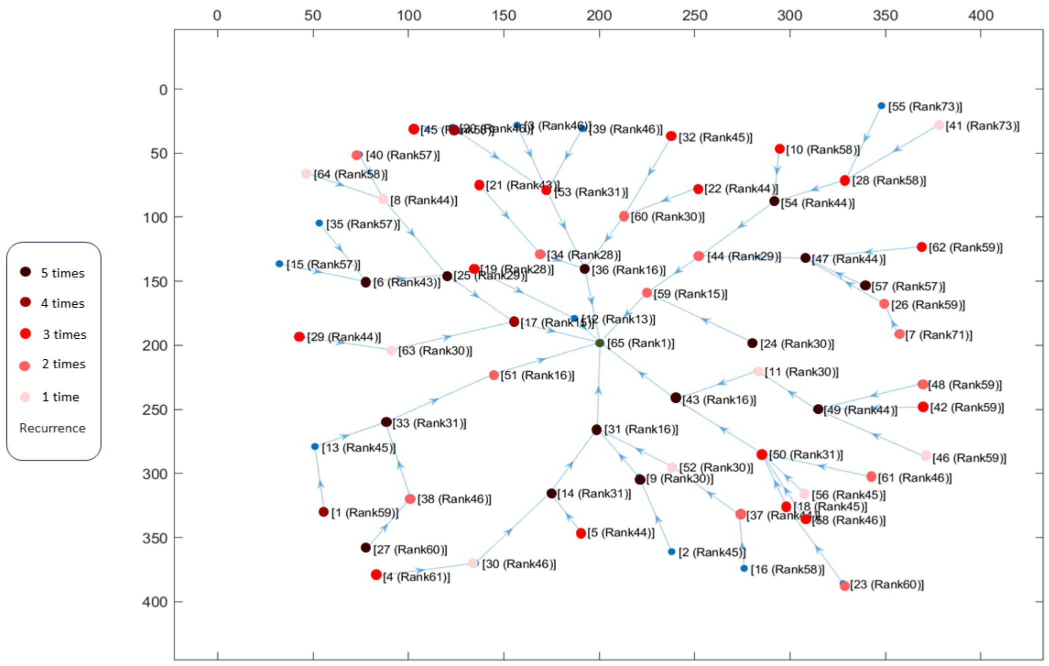


Figure 4.14: *An Illustration of the number of times a node is chosen for probe placement over 5 different runs*

with the flexibility of an optimisation-based heuristic.

The weighting profiles used express a form of relative importance. Security management will need to experiment with different weightings and decide for themselves which outcomes they prefer. Our approaches, however, will furnish management with high-performing outcomes given specific input weighting profiles.

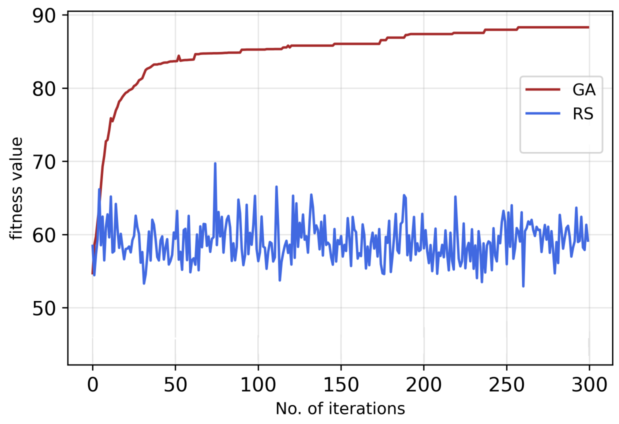


Figure 4.15: *A Comparison between the Random search and Genetic Algorithm on finding optimal candidate solutions. (64-random nodes with* $w1 = 0.7$*(detection rate),* $w2 = 0.1$*,*$w3 = 0.1$*,*$w4 = 0.1$*).*

For illustrative purposes we have chosen to fix certain aspects of the problem, e.g. we have adopted global thresholds for rule threshold values. These constraints can readily be lifted, i.e., node-specific thresholds can be evolved for the rules. It would be prudent to position such node-specific thresholds alongside corresponding node information in the representation shown in Fig. 4.4.

Additional objectives can be adopted. The ones given here are for illustrative purposes; they are plausibly motivated, but we do not maintain they are always the best for specific system circumstances. Detection rate, for example, is an important metric for IDS. However, there are other plausible evaluation criteria for binary classifications; detection rate is an IDS form of what is referred to as “recall” in machine learning, but “False Positive Rate (FPR)”, “False Negative Rate (FNR)” “accuracy”, “Area Under the Curve (AUC)”, and others are all reasonable evaluation objectives that could readily be adopted. Likewise, alternative or further cost-related objectives can be incorporated. Such flexibility is a major strength of optimisation-based approaches.

Finally, although we have been concerned solely with *intrusion detection*, we are aware that aspects of *intrusion response* configuration may also be targeted by our optimisation-based framework. We recommend further consideration of optimisation-based approaches to system configuration to the research community.

4.10 Summary

In this chapter, we presented an optimisation framework based on a Genetic Algorithm for IDS configuration on RPL networks. These networks are now of major importance in many sectors. We have investigated the trade-offs between detection rate, F1 score, coverage, feasibility, and deployment costs. We investigated a single-objective weighted sum approach. We have also investigated both fixed and variable numbers of probes. The performance of the framework has been evaluated using extensive simulations with different networks, weights and numbers of monitoring nodes. Our framework can determine high-performing (and potentially optimal) configurations. The framework is also flexible; incorporating further objectives or dropping current ones will often be easy. The approach does, of course, assume that there is some underpinning notion of a *gradient* in the objective function. For many useful objectives this is the case.

Overall, the work shows that a major network administration task can be effectively automated, achieving levels of performance unattainable by manual means, and freeing the administrator to focus on critical higher-level tasks.

Chapter 5

Fitness Approximation of IDS Fitness Evaluation

In Chapter 4, we built a framework to help network administrators optimise IDS configurations based on a set of useful objectives. The framework is extendable and further desired objectives can be included. However, obtaining optimal (or near-optimal) configurations is time-consuming and computation-intensive. In this chapter, we aim to reduce the complexity by adopting a fitness approximation technique (i.e, surrogate modelling).

5.1 Introduction

As discussed in Chapter 4, developing high-performing IDS configurations in the presence of conflicting objectives is a hard task, particularly in the context of resource-constrained networks. An optimisation-based approach such as an Evolutionary Algorithm (EA), more specifically the Genetic Algorithm (GA), is a natural avenue to pursue. However, and as stated in [105], deploying an EA to complex optimisation problems is far from straightforward. The fitness evaluation of a huge number of candidate solutions before converging to an optimal solution is computationally very intensive and time-consuming.

The evaluation of the fitness function for determining high-performing IDS configuration may be computationally highly intensive. Our research aims to find an efficient proxy for the fitness evaluation that is far less computationally intensive, yet still usefully accurate. In this chapter, we investigate Neural Networks (NNs) for such a purpose. (Technically we seek to use NNs as function approximators for the fitness function used.) The research questions we address are:

- Can optimisation-based approaches to search for high-performing IDS configurations be made more computationally efficient by the adoption of neural networks as fitness approximators for the fitness functions they use?
- To what extent can fitness approximation models used for fitness evaluation be accurate and efficient?

Our main contribution is the development of a fitness approximation model (also known as a meta-model [122]) based on an Artificial Neural Network (ANN), specifically the Feed-forward Neural Network (FNN). This model allows us to radically reduce the time and computation complexity to produce near-optimal IDS configurations. We will use multiple runs of a GA-based approach to solve this problem. Initially, we use a simulator to provide the cost function. The individuals (configurations) in the populations that arise during these runs are evaluated and recorded along with their fitness values. These are used to train the surrogate fitness model. Once trained, this proxy fitness function can be used directly in GA runs, replacing the actual simulation-based approach.

We refer to the GA system using the simulator as GA-Sim (as illustrated in Figure. 5.1), and the GA system with the proxy as GA-FNN. The latter is vastly faster than GA-Sim (as we shall show). Of course, the fitnesses obtained via GA-FNN are approximate. Generally, we aim to find better solutions more quickly. GA-FNN can be used in different ways practically. It can be used to generate high-performing configurations that can be used directly in the system of interest. It can be used to generate a set of individuals with high approximate fitness that can then be evaluated using the simulator (time-wise equivalent to an evaluation of an initial population using GA-Sim). The final population of a GA-FNN run can also be

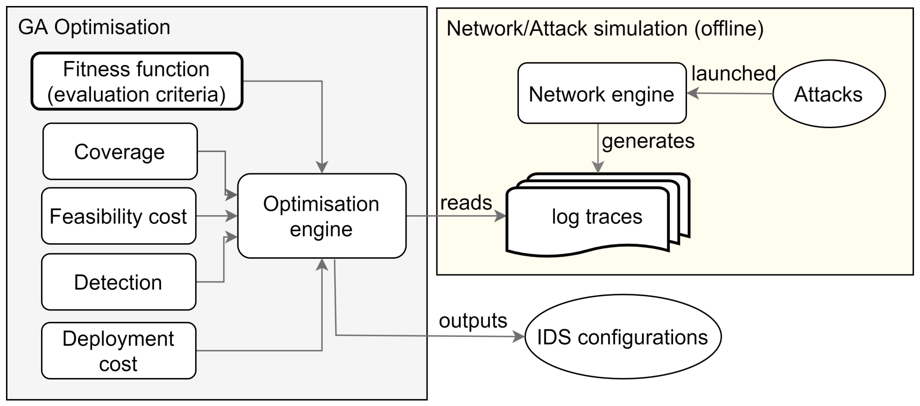


Figure 5.1: *The GA-Sim evaluation process*

used to seed a run of GA-Sim with a high-performing initial population. The above set of uses is not exhaustive.

The rest of the chapter is organised as follows. Section 5.2 presents some relevant previous work concerning optimising IDS configurations. The procedure taken to build the GA and FNN, and to collect the IDS configurations is detailed in section 5.3. The data sampling technique for preparing the training dataset for the FNN is described in section 5.4. Experimental results are given in section 5.5. Discussions and some future insights are given in section 5.6. Section 5.7 concludes the chapter.

5.2 Related work

The literature, as detailed in Chapter 2, proposes various effective approaches to finding optimal configurations of IDS, but they still suffer from significant computation cost issues (which will ultimately limit their usefulness as systems scale up). Below, we propose a computationally efficient approach for optimising IDS configurations.

5.3 System detail

In this section, we will detail the GA-based framework and the NN-based surrogate model. The former is used to optimise the IDS configurations based on set valuable objectives. The generated candidate configurations together with their evaluated fitness measures are then used to train the latter, i.e., the NN model.

5.3.1 GA-based IDS configuration optimisation

The building blocks of the framework (i.e., the dataset generator), which is based on a genetic algorithm to optimise the IDS configurations based on four objectives, are detailed in Section 4.6.

Fitness function: This is designed as a single scalar fitness function (to be maximised). It is presented in the following equation.

$$fitness = ((W_1 \times fit_1) + (W_2 \times fit_2) + (W_3 \times (1 - fit_3)) + (W_4 \times (1 - fit_4))) \times 100 \quad (5.1)$$

It is convenient to scale all objective results by a factor of 100, as shown in Equation 5.1. (Other factors could easily be used.). We assign W_1=0.5, W_2=0.15, W_3=0.15, and W_4=0.2. These are the weights used in this chapter and are given for example only. Different system analyst priorities will give rise to different weights.

The framework generates thousands of IDS configurations where some have high fitness and others do not. As illustrated in Figure. 5.1 and Figure. 5.2 (phase 1), each candidate configuration is evaluated against log traces of network packets generated during simulation. These are processed to provide a fitness evaluation. These configurations are in binary formats and represent the nodes (i.e., possible locations of the IDS probes), which detection rules are deployed at each node and the threshold values used in the rules, as illustrated in Figure. 4.4. Along with their fitness values, all candidate solutions are logged to be used in our dataset. The GA-Sim framework is computationally intensive and time-consuming. To run the experiments, we used HPC facilities at the University of Sheffield.

Hyper-parameters of the GA: The GA runs for 100 generations, with a population size of 100 candidate configurations (bitstrings). It has the standard mutation and crossover routines (see Table 5.1). It uses a 3-tournament selection (i.e., with 3 candidates in each

Table 5.1: *Genetic Algorithm hyperparameters*

Parameter	**Value**
Population size	100
Population type	Bit strings
Number of Generation	100
Selection method	3-Tournament
Crossover method	Two-point
Crossover probability	0.9
Mutation method	Bitflip
Mutation probability	0.01

tournament) with fitness value computed using Equation 5.1. The full list of parameters is given in Table 5.1. The evolution will continue until a termination criterion (in our case, completion of 100 generations) is met.

5.3.2 FNN based fitness approximation

To reduce complexity, we have developed a Feedforward Neural Network (FNN) model to approximate the actual fitness function mentioned in Equation 5.1 which is used in the genetic algorithm framework to optimise the IDS configurations. The neural networks method, in general, and FNN in specific, is chosen as it is known for its accuracy in approximating any nonlinear function [19]. The network's learning rule is to utilise the steepest descent approach to fine-tune the network's weights and biases via backpropagation to minimise the error. In regression problems different loss functions are available, e.g. Mean Square Error (MSE) or Mean Absolute Error. A variety of optimisation methods may be used to manage the process, e.g. Adam or Stochastic Gradient Descent. The learning process of the neural networks is shown in Figure. 5.2 (phase 2). Firstly, the signal is transferred from the input layer to the output layer via the hidden layer, and the difference between the true value and the predicted value is calculated. Secondly, the output error is then transmitted from the output layer to the input layer (i.e., backpropagation), and the weights of the neurons are modified in accordance with the error. Finally, the new weight is applied to the signal and the error is calculated again. This process is repeated until a satisfactory minimised error is obtained or the number of epochs (iterations) is reached.

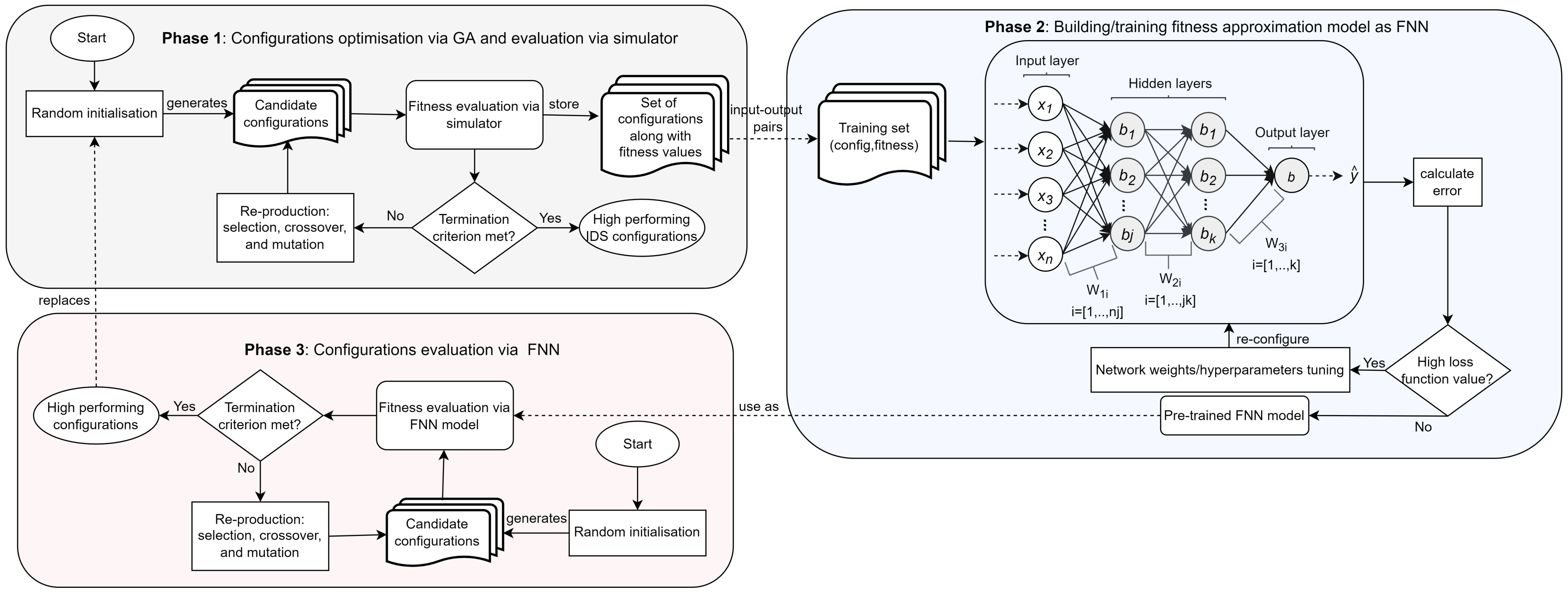

Phase 1: Configurations optimisation via GA and evaluation via simulator
Start
Random initialisation
generates
Candidate configurations
Fitness evaluation via simulator
store
Set of configurations along with fitness values
Termination criterion met?
No
Yes
Re-production: selection, crossover, and mutation
High performing IDS configurations
input-output pairs
replaces
Phase 2: Building/training fitness approximation model as FNN
Training set (config,fitness)
Input layer
Hidden layers
Output layer
x_1
x_2
x_3
x_n
b_1
b_2
b_j
b_k
b
$\hat{y}$
W_{1i} i=[1,..,nj]
W_{2i} i=[1,..,jk]
W_{3i} i=[1,..,k]
calculate error
High loss function value?
Yes
No
Network weights/hyperparameters tuning
re-configure
Pre-trained FNN model
use as
Phase 3: Configurations evaluation via FNN
Start
Random initialisation
generates
Candidate configurations
Fitness evaluation via FNN model
Termination criterion met?
Yes
No
Re-production: selection, crossover, and mutation
High performing configurations

Figure 5.2: *The conceptual architecture of the IDS configurations fitness approximation framework*

Table 5.2: *The Neural Network hyperparameters*

Parameter	Value	Parameter	Value
Input neurons	280	Optimiser	Adam
Hidden neurons	190	Activation function	ReLU
Output neurons	1	Output activation function	Linear
Dataset size	30,000(+5,999 sampled)	Layers	3
Training set	80 % (20% validation)	Epochs	1000
Testing set	20%	Learning rate	0.01
Batch size	32	Cost/loss function	MSE

Phase 2 and 3 (in Figure. 5.2), which are the aim of this research, are dependent on phase 1 where we have to generate a great number of IDS configurations along with their fitness values to be used as a dataset for training the FNN. Then the pre-trained FNN model is used as a fitness approximation (i.e., meta-model) to replace the use of the network simulator as fitness evaluation.

Hyper-parameters of the FNN model: We use Keras library v.2.7 [123] to develop the neural networks model. It is a Python-based sub-library inherited from the TensorFlow backend. The optimal number of neurons and learning rate is chosen by the KerasTuner hyperparameter tuning library [124]. Several search techniques are implemented in KerasTuner and we adopt the Random Search method. We employ the ReLU activation function for the hidden layers as it is the most commonly utilised activation function for modern neural networks [125]. As this is a regression problem, we use the linear activation function for the output layer. We adopt the Mean Square Error (MSE) as the loss function, but we keep track of the other evaluation metrics as will be demonstrated in section 5.5. The complete list of the hyperparameters is shown in Table 5.2.

5.4 Dataset preparation

We run the GA-Sim twice (with different seeds) with a generation and population size of 100. During the genetic algorithm evolution, we log the candidate configurations at each generation

Algorithm 14: Inverse sampling for balancing a dataset based on bins

```
  Input: $dataset_{imbalanced}, B, B_{width}$
  Output: $dataset_{semi-balanced}$
1 begin
2     $dataset_{imbalanced}.sort()$;
3     $dataset_{imbalanced}.divide(B, B_{width})$;
4     $freq_i \leftarrow size(b_i)$;
5     for $i = 1$ to $D$ do
6         Randomly select bin $b_i$ ;                          ▷ Based on $P_i$
7         if $freq_i < n$ then
8             /* n decided experimentally */
              $dataset_{imbalanced} \leftarrow$ randomly uniformly select from within $b_i$
```

along with their fitness values. In other words, the GA-based dataset, which includes 20,000 candidate configurations, is taken from the populations of the genetic algorithm. The key optimisation technique of the algorithm is to keep the best solutions to move on to the next generation; this is called the "survival of the fittest". Consequently, we obtained a skewed dataset where many solutions in the later generations exhibit little improvement in fitness value (i.e., the target variable in our dataset). The dataset, as can be seen in Figure. 5.3 (A), is highly negatively skewed. Consequently, the model may lean more toward the more highly represented instances; and this leads to poor model performance [126].

There have been several methods to deal with skewed datasets in the machine learning community. One of the common techniques for solving imbalanced target distribution is called SMOTE (Synthetic Minority over-sampling Technique) [127]. However, this is intended for classification problems and not for regression problems. Log transformation and square root transformation are other methods, but they usually work well with positive-skewed label problems. We are also aware of the other techniques such as SMOTER and SMOGN [128], Cox-Box [129], but implementing them did not provide satisfactory results.

Therefore, we built a technique based on inverse sampling or inverse probability transformation to enhance the distribution of the samples. We intend to oversample the low represented instances (in classification problems this is called the minority class) and undersample the highly represented instances (in classification problems this is called the majority

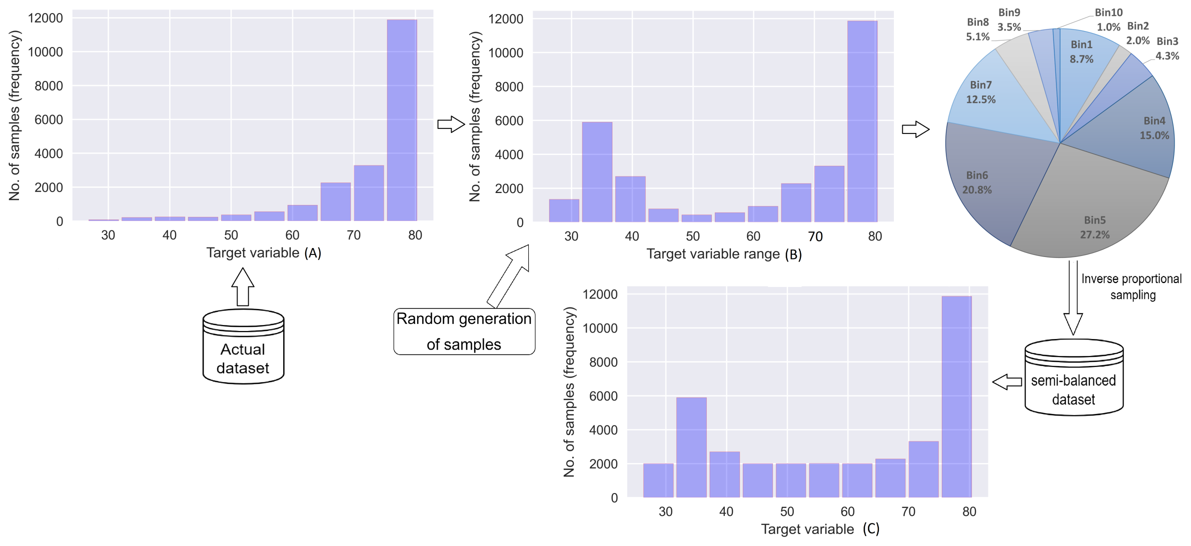


Figure 5.3: *The process of balancing the GA-based dataset*

class) to create a balanced dataset. We divided the dataset into different bins and calculated the probability of selection based on the frequency of each bin. Thus, we select inversely according to the frequency of fitness values of each bin. For instance, let B is the number of bins and D is the number of samples on the intended dataset, $\forall b_i \in B$, the ratio of how often bin b_i is sampled is calculated as $\frac{1}{freq_i}$, where $freq_i$ refers to the number of samples of d_i in b_i. The probability of each bin is then calculated as the following equation:

$$P_i = \frac{\frac{1}{freq_i}}{\sum_{j=1}^{B} \frac{1}{freq_j}} \tag{5.2}$$

We adopted 10 bins with equal fitness ranges (i.e., bin's widths) as shown in Figure. 5.3 (A). The number of instances is not uniform across bins. Thus, some fitness ranges are more highly represented than others. Such imbalances may lead to a poorly trained FNN. Accordingly, we have chosen a biased sampling regime for the generation of the dataset to be used for FNN training. To select which instance to be sampled back to the dataset, we first select a bin with a probability inversely proportional to the number of instances that bin has, and then select uniformly from the selected bin. We only select from bins, b, with $freq$ <2000. The procedure is demonstrated in Algorithm 14. The reason behind this threshold value is that the sampling technique based on bin probability creates many duplicate instances from those bins with low $freq$ trying to balance the bins. However, this did not allow successful training of the model. Therefore, the sampling from any bin is based on a threshold. Our approach mixes a strict inverse sampling regime (to some extent) with keeping some samples (i.e., those in a bin that is above the threshold n) to avoid loss of information on the high-performing solutions. Accordingly, the distribution of the dataset is slightly enhanced as shown in Figure. 5.3 (C).

Furthermore, in order to include candidate configurations not generated by the GA-based approach, we randomly generated $10,000$ evaluated configurations and combined them with the GA-based dataset (see Figure. 5.3 (B)). Note random search alone could not reach near-optimal or optimal candidate solutions as obtained by the GA. This has been illustrated in Section 4.7.2. Having low- and high-performing samples is important to train a model effectively.

5.5 Experiments and results

We applied the genetic algorithm IDS configurations optimisation framework on a 64-node random network. We carried out the network simulation using NetSim v12.1 [120]. We used the university HPC to run the GA-Sim framework to generate the set of IDS configurations. We run the experiments of building, training and using the FNN on a Windows machine with the following specifications: Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz, 3000 Mhz, 8 Core(s), 8 Logical Processor(s), and with 16 GB of RAM available.

5.5.1 FNN model evaluation

The dataset (i.e., a set of IDS configurations) is saved as a CSV file to train the FNN model. We use Mean Square Error (MSE) as the loss function for training but also provide the results of evaluating the model on the most common evaluation metrics for regression problems. The IDS configuration-based FNN model has demonstrated excellent performance as reported in Table 5.3 and visualised in Figure. 5.4.

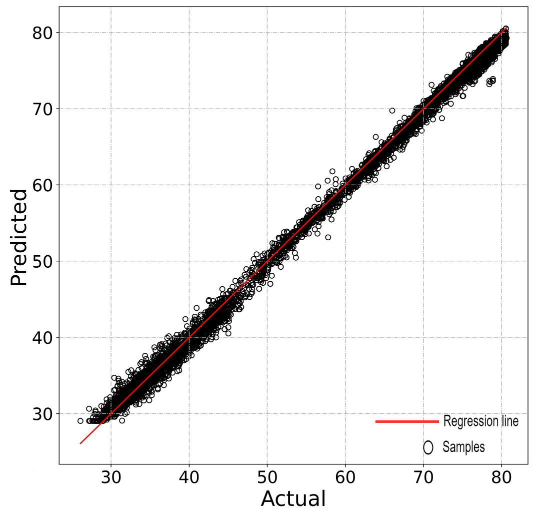


Figure 5.4: *The FNN model evaluation.*

Table 5.3: *Performance evaluation of the IDS configuration-based fitness approximation neural network model*

Metric	value
Mean Square Error (MSE)	1.003
Root Mean Square Error (RMSE)	1.002
Mean Absolute Error (MAE)	0.788
Mean Absolute Percentage Error (MAPE)	0.015
Coefficient of Determination (R^2)	0.997

5.5.2 FNN as fitness evaluation

The work presented in this chapter investigates the effectiveness of using the neural network as a fitness approximation meta-model to reduce the time and computation required for the traditional optimisation techniques when optimising the IDS configurations. We have studied two approaches:

Approach 1: We can now replace the costly fitness evaluation function with the model. We run the GA with the pre-trained FNN model (GA-FNN) as a fitness evaluation, as shown in Figure 5.2 (phase 3). Table 5.4 includes examples of 10 configurations taken from the last population of the GA-FNN. We demonstrate the significantly reduced time for evaluating a single configuration using the FNN model with a small error. On average, the time taken to

Table 5.4: *The difference between the GA-FNN and GA-Sim (in accuracy and time) for evaluating ten configurations*

Samples	GA-FNN*	Time*	GA-Sim*	Time*	Error	Time ratio
config1	85.7	0.026	81.1	30.0	4.6	1153:1
config2	85.5	0.028	81.2	29.0	4.3	1035:1
config3	85.9	0.028	81.4	29.0	4.5	1035:1
config4	85.8	0.026	81.4	29.0	4.4	1115:1
config5	85.9	0.026	81.4	29.0	4.5	1115:1
config6	85.9	0.028	81.4	29.4	4.5	1035:1
config7	85.9	0.027	81.4	29.0	4.5	1074:1
config8	84.0	0.026	81.1	29.3	2.9	1126:1
config9	85.9	0.030	81.4	29.0	4.5	966:1
config10	85.2	0.028	81.4	28.9	3.8	1032:1

* Fitness value. *Evaluation time (in seconds).

evaluate one configuration and retrieve the fitness value via the simulator is ≈ 29.2 seconds while via the FNN is ≈ 0.027 seconds. The ratio of the reduced time of evaluating each single configuration can be represented as ≈ 1:1081 times.

Furthermore, we can find that the GA-FNN model, running for 200 generations and 200 populations, was able to find high-performing configurations (i.e., fitness value = 81.4 as can be seen in Table 5.4 compared to the highest score achieved over 5 averaged extended (500 generations) runs of the GA-Sim = 80.6 as shown in Figure 5.5 (A)). Note that, the approximation-based scores here systematically overestimate the performance but the corresponding configurations are still high performing when evaluated using the simulator. The approximation is simply a means to an end. Of course, we have had to invest a considerable amount of time in generating test data to develop the approximation function. The overall cost required to obtain this high fitness value is calculated as the following: the 30.000 candidate configurations (i.e., the dataset), generated via the GA-Sim and random, cost ≈ 30.000 × 29.2 seconds (on average) = 243.3 hours. The time required to train the FNN model is 4.6 hours. Running the GA-FNN algorithm costs ≈ (200 × 200) × 0.027 seconds (on average) = 18 minutes. Therefore, the total running time is ≈ 248.2 hours. On the other hand, the GA-Sim could not reach the high fitness value obtained by the GA-FNN even after 500 iterations; this costs ≈ (500 × 100) × 29.2 seconds = 405.6 hours. **Showing that the**

Algorithm 15: The implementation of GA-based FNN/Sim fitness evaluation

```
   Input: $n_{gen}, n_{pop}, n_{gene}, cross_{rate}, mut_{rate}$
   Output: $Best\ configuration$
1  // The GA initialisation and detailed process follow what in Algorithm 13
2  begin
3      $last\_pop[]$ ;                                        ▷ Record last pop configurations
4      $g \leftarrow 0$
5      $initialise\_config(P_g)$
6      $evaluate\_via\_FNN(P_g)$
7      while $n_{gen} < g$ do
8          $P_{parent(g+1)} \leftarrow P(g).selectParent()$
9          for $c = 1$ to $(n_{pop}/2)$ do
10             $P_{child(g+1)} \leftarrow Crossover(P_{parent(c)}, P_{parent(c+1)})$ by $cross_{rate}$
11             $P_{child(g+1)} \leftarrow Mutate(P_{parent(c)}\ and\ P_{parent(c+1)})$ by $mut_{rate}$
12     $P_{g+1} \leftarrow creatNextGeneration(P_{child(g)})$
13     $g \leftarrow g + 1$
14     $last\_pop.add(g_{n_{gen}})$;
15     $/ * Next\ start\ the\ GA - Sim * /$
16     $g \leftarrow 0$
17     $last\_pop(P_g)$ ;                                     ▷ configurations from above
18     $evaluate\_via\_sim(P_g)(5.1)$;
19     $/ * Same\ evolution\ (while\ loop)\ process\ as\ above\ * /$
20 return($Best\ configuration$)
```

GA-FNN is faster by a factor of over 1.6.

Approach 2: We extended this work by combining the two evaluation approaches (i.e., the GA-Sim and GA-FNN). This was done by using the FNN-based model as the first evaluation method (i.e., a surrogate model). The last population, which is filtered and obtained by the FNN model, is used as the initial population of the GA-Sim method using the true fitness evaluation. Algorithm 15 demonstrates the process. This will have two benefits. One is that we will obtain true (simulator-based) optimal solutions (rather than optimal as judged by the approximated fitness). Second, the low-performing candidate configurations will be quickly filtered out, and this will result in reducing the number of configurations evaluated. The GA-Sim will start from a much better population rather than being random. Figure 5.5 (B) illustrates the much better starting point for the GA-Sim. Therefore, it was able to reach new optimal solutions in fewer generations. We have run the GA-Sim and GA-FNN five times and taken the average maximum fitness value of each generation.

As far as time is concerned, the time taken to form the initial population using GA-

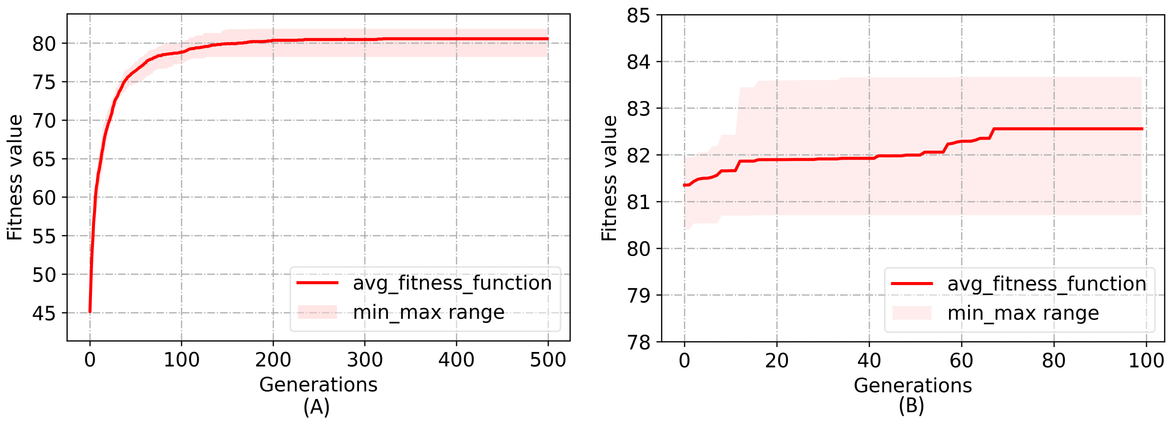


Figure 5.5: *The number of reduced evaluations to reach a high fitness value (on average) between (5 runs of) the GA-Sim with a random initial population (A) and initial population produced by GA-FNN (B)*

FNN, with 200 generations and 100 populations (9 minutes), is ≈ 248.1 hours (as detailed in approach 1). Running the GA-Sim (with 100 generations and 100 populations) consumes ≈ 10.000 × 29.2 seconds = 81.1 hours + 248.1 hours = 329.2 hours. On the other hand, the whole GA-Sim consumes 50.000 × 29.2 seconds ≈ 405.6 hours.

5.6 Brief discussion

The evaluation of a solution via a meta-model is faster than evaluation using simulation software because it involves the use of surrogate models or approximations that can provide an estimate of the system's behaviour. Surrogate models are typically simpler and computationally less expensive. They capture the essential features and behaviour of the system while sacrificing some level of accuracy. This simplification allows for faster evaluations compared to running the full-scale simulations. There are several reasons why Machine Learning (ML), and Neural Networks in particular, can make provide faster predictions than simulations:

- ML models are trained on large data samples to learn patterns and relationships within the data. Once the training is complete, the model is optimised and ready for inference. During making the evaluation, the model utilises the learned parameters and structure to make predictions efficiently. The prediction process often entails performing a series of fast mathematical operations, such as calculations within neural network layers, which can be executed quickly.

- Many ML frameworks and libraries are designed to take advantage of parallel processing capabilities of modern hardware, such as multi-core CPUs or GPUs. These frameworks can efficiently distribute the computational workload across multiple cores, allowing for faster prediction times.
- ML frameworks and libraries often provide optimised implementations of common algorithms and operations used in ML models. These implementations are usually highly optimised and fine-tuned to take advantage of hardware-specific features, such as vectorised instructions or GPU acceleration. This optimisation significantly speeds up the prediction process.

The above characteristics of surrogate-models make the evaluation of a single IDS configuration 1000x faster than a simulation-based evaluation.

In our case, we are motivated in significant part by a desire to allow configurations to change regularly. This is needed when 'fingerprinting' of the system is perceived to be a problem. If attackers can learn the configuration of the IDS system they can exploit that knowledge to their advantage. Shifting configuration regularly makes such fingerprinting more difficult. *With such multiple uses, the performance enhancements provided by our approach become overwhelming.*

We have adopted straightforward objectives in our work. However, detection strategies may become collaborative and far more complex. In such cases, the complexity of the simulation will increase and so the need for fast approximation will become greater.

Further performance criteria can easily be added. These could include diversity criteria to ensure the generation of configurations is sufficiently different to previously used ones. The evaluation of plausible diversity metrics will likely be computationally insignificant compared to aspects of system simulation.

The GA-FNN is trained to approximate a fitness landscape for one specific 64-node network. As future work, it would be highly desirable if the learning for one particular network can be reused for another. This could radically reduce the overall costs of approximation (the greatest part of which is in generating training data). In the next Chapter we investigate whether *transfer learning*, a well-established concept in Deep Learning, can be harnessed to provide such benefits for fitness approximation.

5.7 Summary

IDS probe placement and configuration is a promising target for optimisation-based approaches. Evolutionary algorithms have been developed, by ourselves and others, that are quite effective at making the important multi-criteria tradeoffs. However, these are always computationally highly intensive and time-consuming.

In this chapter, we have shown how our approximation-based approach GA-FNN is faster (more than 1.6x) and produces better results than the simulation-based GA-Sim. This means that advanced applications, such as regular changing of configuration to avoid fingerprinting, will not be impaired by the speed of optimisation. Furthermore, for such applications, the speed of response (finding an appropriate new configuration) may be more important than squeezing out the last ounce of performance. Either way, the excellent level of performance of GA-FNN will prove highly beneficial.

Approximating complex simulation-based fitness functions using neural networks for generating high-performing IDS configurations has been shown to be feasible, enabling better solutions to be obtained more quickly. We believe the need for further approximations research in this area will grow.

Chapter 6

A Transfer Learning Approach to Discover IDS Configurations

Reducing the computation required for a simulation-based optimisation technique to obtain high-performing IDS configurations for a particular network is achieved in Chapter 5. However, when it comes to new networks, we would then need to run the process again, from scratch. Here, in this chapter, we aim to build a generalised fitness approximation model that can produce high-performing IDS configurations not only for one specific network but also for variant networks.

6.1 Introduction

Runs of the optimisation algorithm are computationally intensive and time-consuming [105]. As networks scale, the resource consumption problem increases. Furthermore, analysts may desire to have multiple runs of an algorithm and compare the results before making a specific choice. It may also be desirable to be able to generate many possible high-performing configurations to allow the IDS deployment to evolve, e.g., where a static configuration might lead to 'fingerprinting'. The major cost in all search approaches is evaluating the performance of a configuration, which for difficult criteria such as detection rate may require a simulation.

The surrogate model has been shown to be effective and efficient in optimising the IDS configurations for a specific network. The generalisation of the approach is still a limitation. The main contribution of this chapter is to build a generalised IDS fitness approximation model using a transfer learning Deep Neural Networks (DNNs) approach. The transfer learning method allows prior experience of evaluation of other networks to provide a more accurate and efficient means of evaluating newly presented networks. First, we use a specific evolutionary-based optimisation method -- a Genetic Algorithm (GA) -- to generate a set of evaluated IDS configurations for various network typologies. We use these as the training set for our DNN model. We test the ability of the pre-trained model to approximate the fitness function of IDS configurations on unseen networks. The evaluation using such a model is extremely fast and can significantly reduce the computational expense of traditional optimisation approaches. The knowledge learned by a model trained on a set of variant networks is transferred to a model trained on a smaller number of networks (i.e., fewer data samples). We show how transfer-learning-assisted DNNs can perform better (with fewer training epochs) than standalone DNNs. This reduces the training time while maintaining high prediction accuracy (or minimum error).

The remainder of this chapter is organised as follows. A brief explanation about the IDS and the targeted attack is presented in Section 6.2. In Section 6.3, we present the fitness function adopted in this chapter, which involves the objectives detailed in Section 4.6. The same framework is used to produce the data for our DNN training. Section 6.4 details the network simulation, the network generator, feature engineering, DNN implementation and the transfer-learning technique. The results are given in section 6.5. Section 6.6 presents a summary of the chapter.

6.2 Rule-based IDS to detect greyhole attacks

In this chapter, we investigate how NIDS can best discover RPL attacks, more specifically the Greyhole attacks (also known as selective forwarding attacks). The IDS sensors monitor their neighbourhood for any malicious activities. We also adopt the rule-based IDS as it is effective in detecting RPL attacks and efficient for resource-constrained networks [130]. The adversary node selectively drops packets; hence, we monitor the Packet Dropping Ratio (PDR) of each node in the network. This is the threshold value and is encoded into a binary format as shown in Figure 6.1. Thus, a rule in our system fires if:

EXIST a node x in an IDS sensor neighbourhood such that PDR(x) $\geq T$ in t seconds.

6.3 Dataset creation

In this section, we describe the building blocks of the optimisation framework that is used to find optimal IDS configurations. These are mainly used to generate datasets to be used to train a deep learning model to approximate a fitness function. This reduces the computation complexity required by the optimisation mechanism. Further, we show how transfer learning can help achieve better results than a standalone deep learning model.

In order to train the DNN model, we need to generate a dataset. Our dataset includes a set of IDS configurations. Each configuration is evaluated against four objectives as described in Section 4.6. These are optimised using a framework based on an Evolutionary Algorithm (EA). (Our dataset needs to include high-performing configurations and an EA optimisation approach is a good way of obtaining them. Randomly generating configurations will not suffice, as shown in Section 4.7.2). Configurations are stored in a binary format and they describe the nodes, probe placement and a threshold value (used to define when a specific rule should 'fire'). Figure 6.1 illustrates the IDS configuration representation.

6.3.1 Fitness measurement

The fitness of each configuration is assessed by its ability to satisfy a set of objectives as detailed in Section 4.6. In this work, we adopt the detection rate for the first objective. Regarding the feasibility cost objective, we have two different types of nodes, low-energy-nodes and moderate-energy-nodes. Thus, this objective will punish (and so discourage)

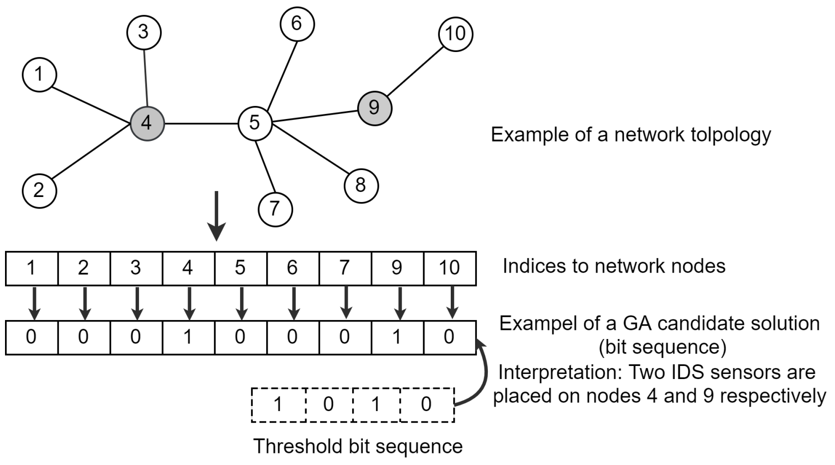

Example of a network tolpology
Indices to network nodes
1 2 3 4 5 6 7 9 10
0 0 0 1 0 0 0 1 0
Exampel of a GA candidate solution (bit sequence)
Interpretation: Two IDS sensors are placed on nodes 4 and 9 respectively
1 0 1 0
Threshold bit sequence

Figure 6.1: *Example of IDS sensor configuration representation.*

placing IDS sensors on the low-energy-nodes. We target only one rule ($R = 1$) hosted by a probe with the aim of detecting selective forwarding attacks. However, the objective is generalised and can be used with a wider set of hostable rules. We therefore either host (place) a probe or we do not. We set the m_{rj} (in Equation 4.6) of the low-energy-nodes to 0 and to 1 for moderate-energy-nodes.

Fitness Function: We adopt a Weight Based Genetic Algorithm (WBGA) [131], formulating a multi-objective problem as a single-objective one. The objective functions are weighted by a vector of scalars (weights). Different system analyst priorities will give rise to different weights. For illustration purposes, we have used the following weighting values: $W_1 = 0.5, W_2 = 0.15, W_3 = 0.15$ *and* $W_4 = 0.2$. The fitness function, which is a combination of the Objectives 4.1 - 4.5, is defined as follows.

$$maximise \rightarrow fitness = 100 \times ((W_1 \times fit_1) + (W_2 \times (1 - fit_2)) + (W_3 \times fit_3) + (W_4 \times (1 - fit_4))) \tag{6.1}$$

We note that each objective f_j can reach an optimum of 1 or 0, depending on whether it is to be maximised or minimised. It is convenient to scale all objective results by a factor of 100, as shown in Equation 6.1. (Other factors could easily be used.) Since the weights are sum to 1, the overall fitness function is seen to lie between 0 and 100. However, because of the interaction between the objectives, we cannot know for sure what the achievable maximum is.

6.4 Experimental set-up

6.4.1 Network simulation

We use Netsim v12.1 [120] to simulate our experimental networks. The whole network consists of 32 wireless nodes. These nodes differ in terms of available energy. 20% of the nodes are low-energy and the rest are moderate-energy. All nodes generate two types of packets: the control message packets, which are used to construct the DODAG topology and connections, and application packets. The radio range between the nodes is set to 78 dBM which means they can communicate within a distance of $\approx$ 50 meters. We have experimented with each node being attacked for the sake of finding the best placement wherever an attacker node happens to be.

6.4.2 Network generator

To evaluate the DNN-based approximator on different networks, we designed a Random Network Generator (RNG). This takes as an input a configuration file (xml format) of a specific network and generates random topologies of the same network size (see Figure 6.2). In total, we generated 33 random networks. Nodes can be placed on any location on a 200 x 200 meter plane (except the border router which is fixed in the centre). The RNG takes into consideration an important condition, which is that the nodes are not closer than a predefined minimum distance (i.e., distance $>$ 10 meters). In a Wireless Sensor Network (WSN), if the radio ranges of two or more nodes are significantly overlapped and they transmit or broadcast packets at the same time, this may cause packet collision [132]. To maintain the connectivity between the nodes, each node should be within range of at least one node to avoid any node being disconnected from the network. This is checked via the network simulation.

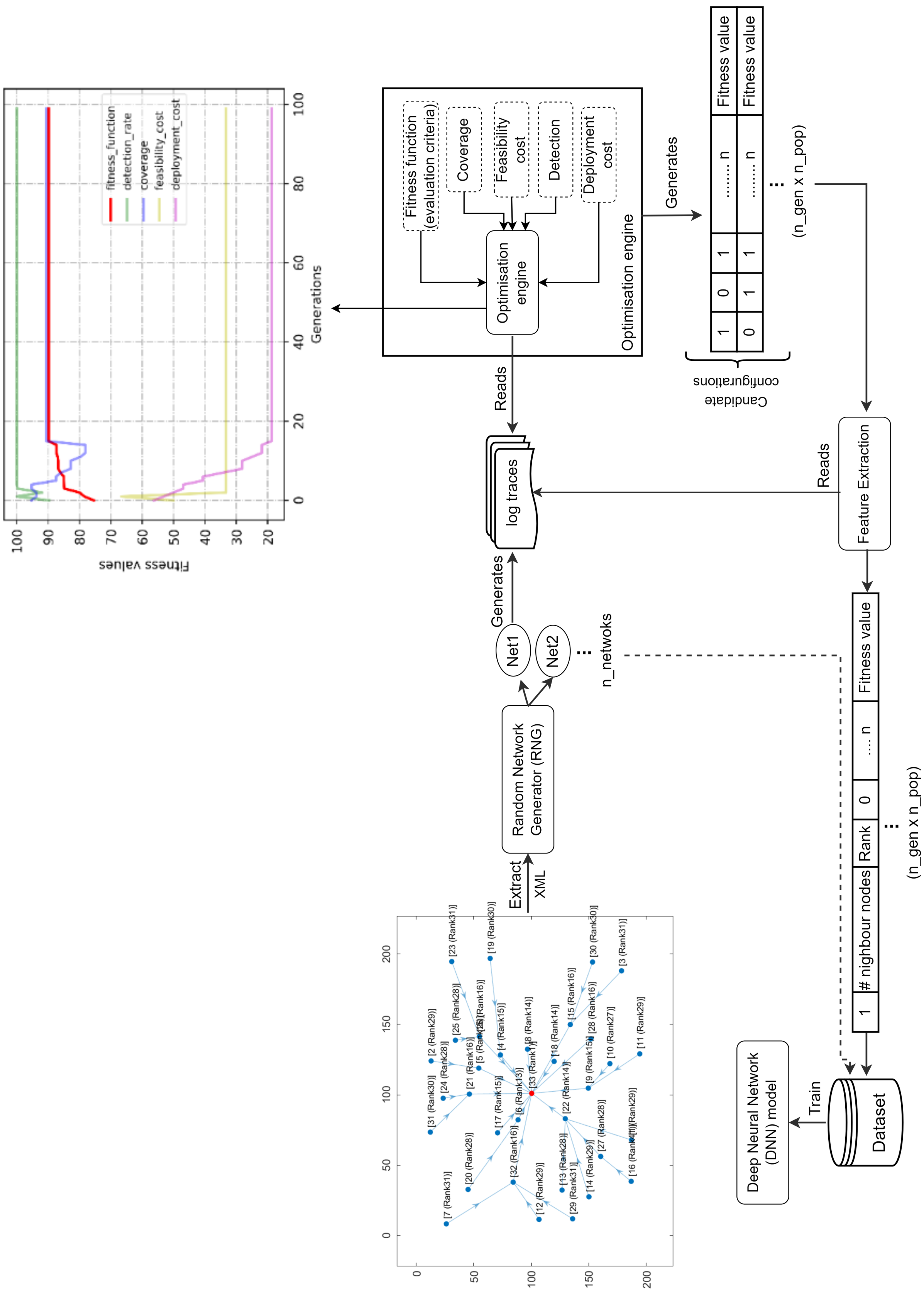


Figure 6.2: *High-level overview of IDS configurations optimisation, generation and preparation.*

6.4.3 Feature engineering

We have extracted two main features that were important factors when optimising the IDS deployment for RPL networks. These are the number of neighbouring nodes and the rank of each node. The rank of each node is calculated based on the location of a node from the border router (6BR). The lower the rank, the closer to the 6BR. Thus, the rank indicates the location of the node in the DODAG.

The current configurations are in a binary format. However, the number of neighbouring nodes and the ranks are integer inputs. Thus, to avoid the model being impacted by these large values, we normalised the extracted features (i.e., min-max scaling between 0 and 1).

Figure 6.2 illustrates the whole process of generating networks, optimising the IDS configurations of each network, extracting the configuration, and feature engineering.

6.4.4 Neural Network building

We use Keras [123] to build the DNN model. It is a Python-based library that is implemented on top of the TensorFlow framework. The whole network configuration is reported in Table 6.1. We adopt the Mean Absolute Error (MAE) as a performance loss function. We recorded other evaluation metrics such as Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE) and R2 square. These are detailed next.

MSE (Mean Square Error). This measures the average of the squares of the errors or residuals. That is, the average squared difference between the predicted value $\hat{y}_i$ and the actual value y_i.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where n is the number of observations.

MAE (Mean Absolute Error). This measures the average of the absolute error or residual values. In other words, the average absolute difference between the predicted value $\hat{y}_i$ and the actual value y_i.

Table 6.1: *The Neural Network hyperparameters*

Parameter	**Value**	**Parameter**	**Value**
No. input neurons	100	Optimiser	Adam
No. hidden neurons	67 (in each layer)	Activation function (hidden layers)	ReLU
Cost/loss function	MAE	Activation function (output layer)	Linear
No. output neurons	1	No. hidden layers	3
Learning rate	0.001%	Batch size	32
Epochs	500		
Dataset size (*source-model*)	270,000	Dataset size* (*target-model*)	10.000
Dataset size†	60,000		
Training set*	varied (20% validation)	Testing set*	varied
Training set†	50 % (20% validation)	Testing set†	50 %

*Experiment 1. † Experiment 2

$$MAE = \frac{1}{n}\sum_{t=1}^{n} \mid y_i - \hat{y}_i \mid$$

RMSE (Root Mean Squared Error). While MSE measures how far the predicted values are from the regression line (i.e., prediction error), RMSE shows how spread out these errors are.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

MAPE (Mean Absolute Percentage Error). This measures how accurate a prediction model is.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(y_i - \hat{y}_i)}{y_i}\right|$$

R2 (R-squared). This is also called the *coefficient of determination.* It measures the relationship between the dependent and independent variables. In other words, in a regression model, it represents how much variance in a dependent variable is explained by the independent variable(s).

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

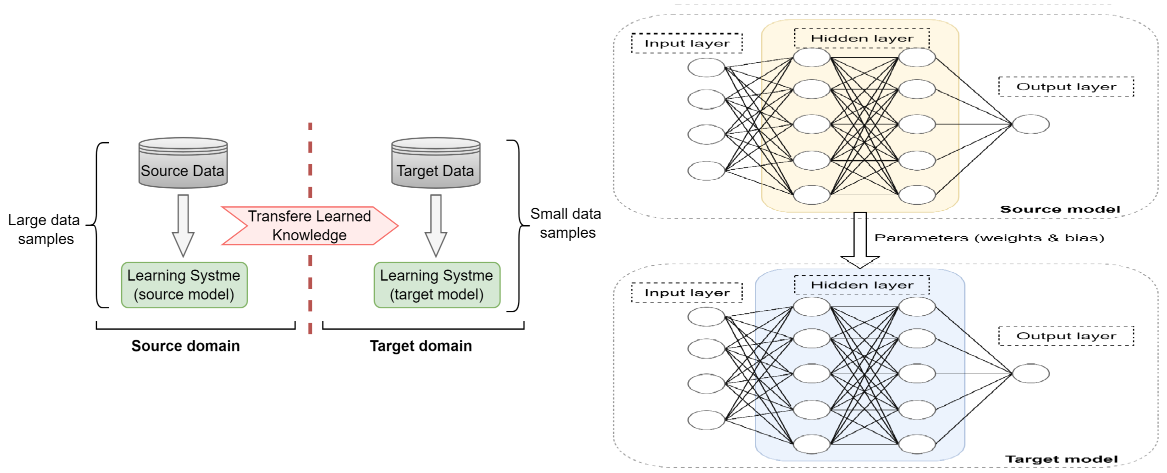


Figure 6.3: *Learning process of transfer learning.*

$\bar{y}$ indicates the mean of the actual data and is calculated as $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

6.4.5 Transfer Learning

In Machine Learning (ML), transfer learning seeks to transfer the knowledge gained from one problem to a different yet related one (see Figure 6.3). The benefit of using such a technique is to reduce network training time while maintaining high accuracy. Deep transfer learning is classified into four categories [133]: instances-based deep transfer learning, mapping-based deep transfer learning, network-based deep transfer learning, and adversarial-based deep transfer learning.

We adopt the network-based deep transfer learning which refers to the re-use of parts of the neural networks of the source domain pre-trained model (including the network structure and parameters) on the target domain deep neural network. As illustrated in Figure 6.3, we transfer the hidden layers' parameters of a neural network model (i.e., source model), that we train on a large set of IDS configurations of many randomly generated RPL networks, into a neural network model (i.e., target model) trained on fewer new RPL networks. Here we hope that with fewer data samples and with the help of transfer learning, we will still obtain an excellent prediction model (i.e., with low error).

Table 6.2: *Genetic Algorithm hyperparameters*

Parameter	Value
Population size	100
Population type	Bit strings
Number of Generation	100
Selection method	3-Tournament
Crossover method	Two-point
Crossover probability	0.9
Mutation method	Bitflip
Mutation probability	0.01

6.5 Results

We have trained a DNN model on a large set of IDS configurations generated from 27 randomly generated RPL networks. For each network, we run the genetic algorithm optimisation to optimise the set of objectives mentioned in Section 4.6. The parameters used during the genetic algorithm process are presented in Table 6.2. We have run our genetic algorithm, with a population size of 100 (i.e., 100 candidate configurations). The population is evolved over 100 generations. This generates 10,000 IDS configurations with a fitness value assigned to each one that represents its goodness in terms of the studied functional and nonfunctional objectives. For 27 training networks (this is from Network 1 - 27 of the total 33 networks), we have obtained 270,000 IDS configuration samples. This forms the dataset used to train the source model. This is the pre-trained model we utilised for the transfer learning method. The aim here is to study and compare the ability of two approaches, namely the standalone and transfer learning DNN approaches, to approximate a fitness function for new variant networks. This is to show the ability of a model that has been trained over a set of variant network configurations to accurately predict the fitness value of new networks. We try to solve a generalisation problem here using a cheap (yet approximated) model to replace the traditional complex and expensive optimisation methods. We study the problem using the two approaches outlined below.

6.5.1 Experiment 1: retraining model vs transfer learning model

As mentioned in Section 6.4.5, the benefit of transfer learning is when there is limited access to training data samples. This is the case with newly presented networks where there is no knowledge/data about the IDS configurations for these networks unless running the expensive optimisation technique. Therefore, we evaluated the performance of the retraining model and the transfer learning model over different dataset sizes. An explanation of each technique is given next.

The standalone (retraining) model: This is a DNN model that is trained from scratch. Meaning, there is no prior knowledge about the problem at hand and the neural network parameters are initiated randomly.

The transfer learning model: Here, we have trained a source DNN model on very large configuration samples of 27 networks (i.e., 270,000 samples). The general process of the transfer learning approach is illustrated in Figure 6.3 and the hyperparameters are presented in Table 6.1. We then transfer the knowledge (i.e., the trainable parameters such as weights and biases) from this model to support a target DNN model during the training. Therefore, these parameters will start from some enhanced values, based on the harvested knowledge of the source model, rather than random.

Both models were trained on different training set sizes of IDS configurations. Here we have access to limited samples to train the models. Figure 6.4 demonstrates the prediction accuracy of the two schemes. The transfer learning based model was able to make a more accurate fitness evaluation than the retraining model over all training set sizes. We trained the models on the specified training sets (i.e., 10%, 20%...etc) and tested on the rest. Furthermore, during the training, the transfer learning approach was able to quickly converge to a low training error (as shown in the right-hand side of Figure 6.4) in a fewer number of epochs than the retraining approach. Note, the usage of the early-stopping technique [134] breaks off the training in fewer epochs than specified (500 epochs); this is used to prevent the model from over-fitting. More error measures are reported in Table 6.3.

Figure 6.5 shows the mean absolute error (MAE) and mean square error (MSE) with

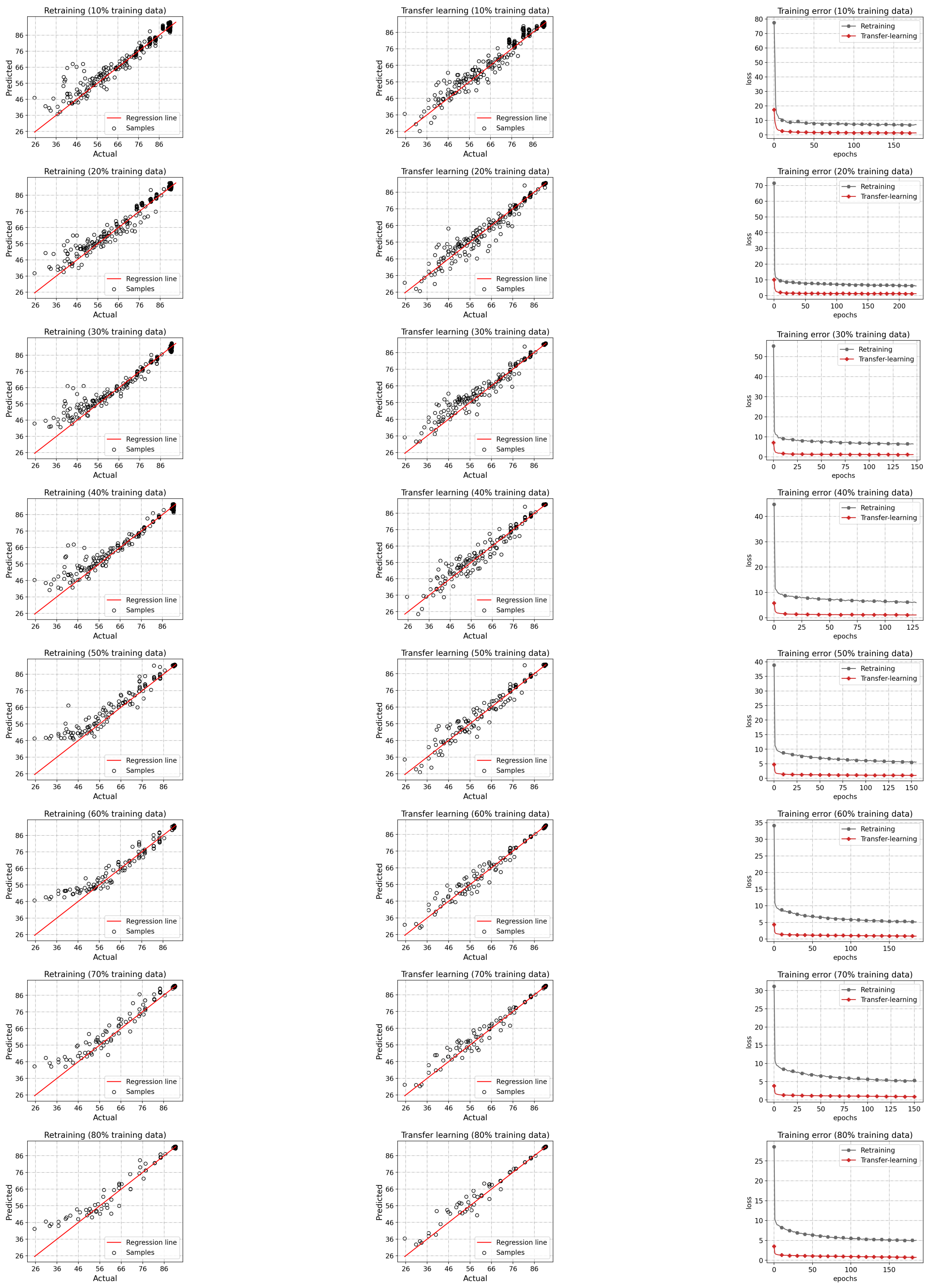


Figure 6.4: *Performance illustration between retraining and transfer learning approaches using different dataset sizes (on Network 28).*

Table 6.3: *Performance evaluation of the retraining and transfer learning approaches.*

	Traning data (%)	Metric	Retraining	Transfer-learning
Network1[†]	10	MSE	1.126	0.66
		RMSE	1.061	0.812
		MAE	0.596	0.24
		R^2	0.948	0.969
		MAPE	0.007	0.003
	20	MSE	0.707	0.408
		RMSE	0.841	0.638
		MAE	0.272	0.09
		R^2	0.97	0.983
		MAPE	0.004	0.001
	30	MSE	0.965	0.465
		RMSE	0.983	0.682
		MAE	0.393	0.1
		R^2	0.962	0.982
		MAPE	0.005	0.002
	40	MSE	0.994	0.442
		RMSE	0.997	0.665
		MAE	0.308	0.102
		R^2	0.964	0.984
		MAPE	0.004	0.002
	50	MSE	0.998	0.398
		RMSE	0.999	0.631
		MAE	0.159	0.093
		R^2	0.963	0.985
		MAPE	0.003	0.002
	60	MSE	0.962	0.299
		RMSE	0.981	0.547
		MAE	0.193	0.077
		R^2	0.964	0.989
		MAPE	0.003	0.001
	70	MSE	0.983	0.37
		RMSE	0.991	0.608
		MAE	0.189	0.089
		R^2	0.967	0.988
		MAPE	0.003	0.002
	80	MSE	0.94	0.472
		RMSE	0.97	0.687
		MAE	0.153	0.091
		R^2	0.972	0.986
		MAPE	0.003	0.002

[†] This is Network 28 of the total 33 networks.

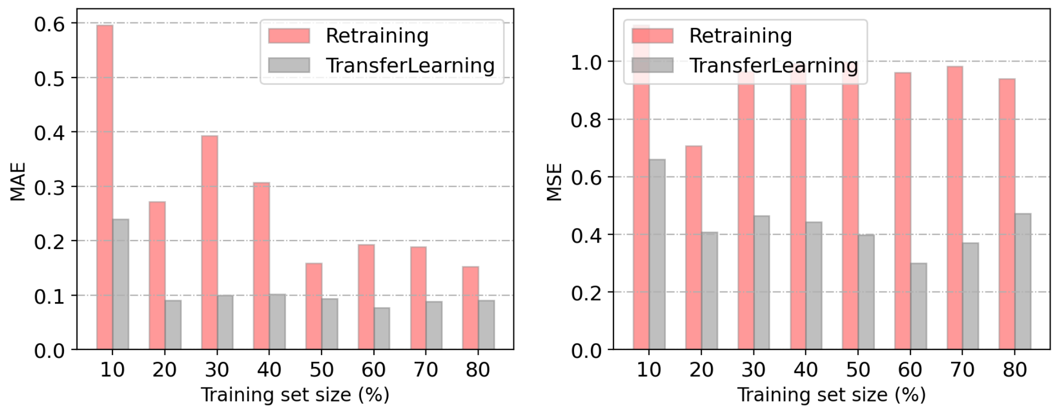


Figure 6.5: *The Mean Absolute Errors and Mean Square Errors of the transfer learning and retraining approaches using different training set sizes (on Network 28).*

respect to the training set size. The errors tend to be reduced as the training set size increases. The mean absolute errors and mean square errors produced by transfer learning are lower than those produced by the retraining approach

6.5.2 Experiment 2: Other ML models vs transfer learning model

To evaluate the performance further, we compared the results of the transfer learning approach (and the retraining model for completeness purposes) with two common machine learning models for regression problems, namely Random Forest Regression (RFR) and Support Vector Regression (SVR). To reduce the confusion of showing many evaluation metric values and illustrations, we evaluated the performance of the models on a dataset that combines the IDS configuration datasets of 6 networks (this is from Network 28 - 33). The models were trained on 50% of the dataset and tested on the other 50%.

The hyperparameters of the transfer learning model (and the retraining model) are the same as presented in Table 6.1. The two important parameters of the RFR model are the number of regression trees and the depth of each tree. We set the former to 500 and the latter to 5 and 10. As a kernel function for the SVR, we used Radial Basis Function (RBF) and polynomial. These kernels are known to be effective for non-linear high-dimensional feature spaces [135]. The evaluation performance is reported in Table 6.4 and Figure 6.6.

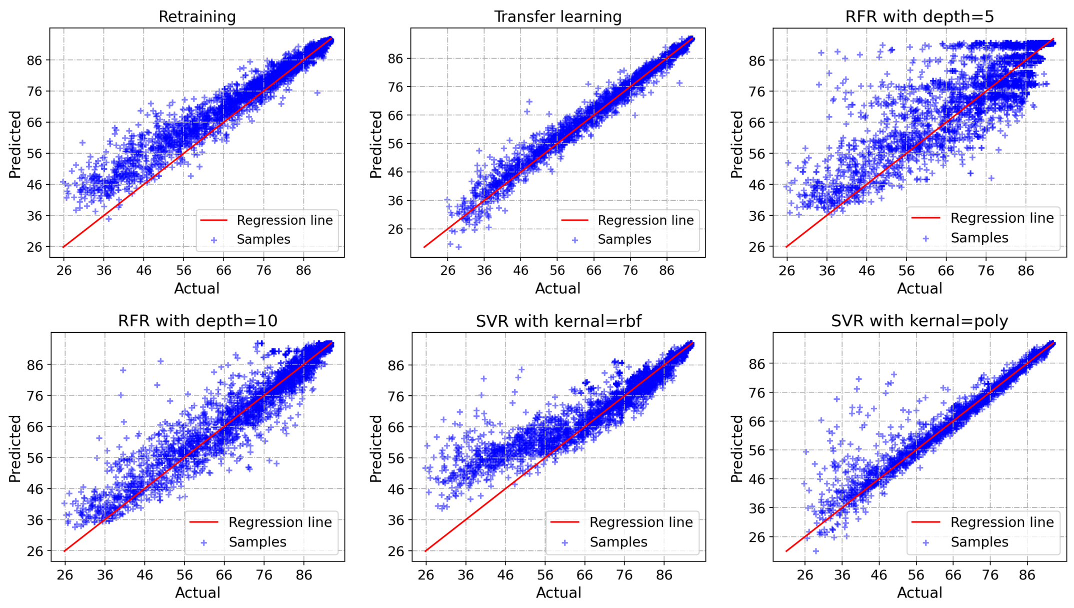


Figure 6.6: *The prediction performance between the transfer learning, retraining, RFR and SVR models on a combined configuration dataset of six networks*

The transfer learning approach outperforms both of them.

Table 6.4: *Performance evaluation of different schemes.*

Metric	**Retraining**	**Transfer-learning**	**SVR**		**RFR**	
			RBF	Polynomial	5*	10*
MSE	3.294	**0.609**	5.148	1.442	9.272	2.896
RMSE	1.815	**0.78**	2.269	1.201	3.045	1.702
MAE	0.562	**0.195**	0.526	0.244	1.361	0.5
R^2	0.947	**0.99**	0.917	0.977	0.85	0.953
MAPE	0.009	**0.003**	0.01	0.004	0.019	0.008

*Depth of the tree.

6.6 Summary

Using evolutionary algorithms for optimising IDS configurations is computationally expensive. Our function approximation model based on deep learning evaluates IDS configurations vastly quicker (and so can be used as a proxy fitness evaluation in optimisation-based searches). However, when a new network requires optimising, the process starts from scratch again. We have shown how a transfer-learning approach achieves better results than a standalone (i.e., from scratch) model for generalising the function approximation for variant networks. Thus, previous experience can be effectively harnessed to efficiently generate high-performing deployments for newly presented networks. The approach can be generalised with the addition of further objectives.

Chapter 7

Conclusions and Future Work

In this PhD thesis, we introduced an optimisation methodology for IDS placement and configuration for the RPL networks based on a Genetic Algorithm (GA). These networks are currently quite significant in many industries. The RPL is designed to make the communication of the nodes efficient. These devices are resource-constrained in terms of power, memory and processing. Deploying a heavy intrusion detection mechanism might not be applicable. Furthermore, the deployment incurs trade-offs between different functional and non-functional criteria. The GA is described to solve the problem. However, IDS configuration evaluation is computationally expensive and time-consuming. Therefore, we investigated fitness approximation as a means to reduce the evaluation complexity. This is called meta-modelling or surrogate modelling. Neural Networks (NNs) are utilised to build fitness approximation models to accelerate the fitness evaluation of the IDS configurations. The approach targets a specific supplied network. However, we wished to exploit previous experience when presented with new networks. This motivated our investigation of transfer learning. The harnessed knowledge of training a model over different networks shows effective outcomes. An evaluation section is provided next to illustrate how the accomplished contributions provide evidence for our research hypothesis.

7.1 Contributions evaluation

The three research hypotheses for this thesis are as follows:

Hypothesis 1: *Evolutionary algorithms can discover resource-efficient and detection-capable security configurations for intrusion detection systems that are suitable for RPL-based Internet of Things networks.*

Chapter 4: RPL networks are resource-constrained, making IDS placement and configuration an especially challenging task. In general, when a task becomes highly complex, there is sometimes no practical alternative to applying some sort of heuristic. We have demonstrated that optimisation-based approaches can be used effectively to address issues of substantial significance in widely adopted and constrained networks. The research provided in this chapter shows that optimisation-based methodologies have a lot to offer. It appears unlikely that any other approach could discover high-performing allocations with the flexibility of an optimisation-based heuristic.

In our genetic algorithm optimisation framework, we targeted four important objectives. These are the classification measure (detection rate or F1 score), coverage, feasibility cost, and deployment cost. We investigated how to find optimal or excellent combinations of these objectives. A single-objective weighted sum technique (i.e., a Weight Based Genetic Algorithm (WBGA)) was developed to solve the problem. We heavily use the weightings of objectives. Different weightings correspond to different aims and priorities and give rise to different outcomes. However, mapping such profiles to de facto notions of importance as judged by analysts remains a tricky task. In practice, security management will need to experiment with different weightings and decide for themselves which outcomes they prefer. Our approach, however, will furnish management with high-performing outcomes given specific input weighting profiles.

Hypothesis 2: *Machine learning approaches can allow us to perform function approximation for the framework's fitness evaluation function and so greatly reduce the time and* *computation taken to produce near-optimal security configurations using such a framework.*

Chapter 5: We demonstrated that using machine learning for fitness approximation is a valid approach to be used as a surrogate model to accelerate the convergence of the genetic algorithm to optimal or excellent IDS configurations. The Genetic Algorithm-based Feedforward Neural Network (GA-FNN) strategy, based on approximation by a FNN, is faster (more than 1.6x faster) and yields better results than the simulation-based GA. This means that the speed of optimisation won't affect complex applications, such as frequent changes of configurations to prevent fingerprinting attacks. It has been demonstrated that it is possible to approximate sophisticated simulation-based fitness functions using neural networks to produce high-performing IDS configurations. This makes it possible to find better solutions more quickly. A great deal of time is expended in generating appropriate data (configurations and their evaluations) on which to train the approximator. If multiple candidate solutions are required by the analyst then this resource consumption can be amortised over such multiple (and possibly a great many) uses.

Hypothesis 3: *A transfer learning based deep neural networks approach can provide a highly efficient fitness approximation with acceptable fidelity for newly-presented RPL-based Internet of Things networks.*

Chapter 6: In this contribution, we generalised the fitness approximation surrogate model. Therefore, we utilised the benefit of a transfer-learning approach to gain knowledge from variant networks and be able to find IDS configurations for other new networks. We showed how a transfer learning DNN outperforms a stand-alone DNN. The technique provides better results (even with variant dataset sizes) with very low error (mean absolute error). Furthermore, we evaluated the transfer learning approach with the most common state-of-the-art machine learning regression models: support vector regression (SVR) and random forest regression (RFR). With different regression evaluation metrics, the transfer-learning still provides the best performance. This shows that the transfer-learning is an effective evaluation approach to produce optimal (or near-optimal) IDS configurations for newly presented networks. We think there will be a growing demand for more studies on approximations in this field.

7.2 Future work

As a future direction, we aim to target the following:

7.2.1 A multi-objective framework for optimising IDS configurations

The problem involves competing objectives and we adopted a weighted-sum optimisation-based approach. We can, however, avail ourselves of various multi-objective optimisation frameworks. One of the widely adopted approaches is called the Non-dominated Sorting Genetic Algorithm (NSGA-II) where objectives are optimised independently and a non-dominated set of solutions is sought, usually referred to as a Pareto Front. (A solution is non-dominated if every other solution in that set has at least one fitness component with worse performance.)

7.2.2 Rigorous evaluation metrics

The detection rate is the prominent evaluation metric for intrusion detection systems. However, we can enrich the framework by making richer use of the underpinning true and false positive and negative measures. The framework is extendable and that allows us to include more functional or non-functional objectives as desired.

7.2.3 Mobile environment

RPL-based networks are currently stationary environments [8]. However, there are some extensions to the protocol to be used for mobile networks. One example is Co-RPL: RPL routing for mobile low-power wireless sensor networks using the Corona mechanism [136]. Our work could be extended by the inclusion of mobile nodes and adopting the framework accordingly to find optimal IDS configurations.

7.2.4 Dedicated sniffers placement optimisation

Rather than hosting the IDS sensors in the networks' nodes, IDS probes could be placed on separate stand-alone nodes that can have more placement options on the network. Of course, this will require more equipment (and thus a greater financial budget) yet the hosting nodes can have more resources available and therefore the ability to perform heavy and more sophisticated intrusion detection mechanisms.

7.2.5 Collaborative intrusion detection

The subject of optimising the configuration of Collaborative IDS sensors is an interesting research topic to target in the future. There are many strategies that could be adopted as to where and how detection decisions are made. Collaborative decision-making is technically more complex and our underpinning simulation-based evaluation may be the only feasible means of evaluation in many cases.

7.2.6 Intrusion Prevention System (IPS)

Although we have been solely concerned with intrusion detection, we are aware that aspects of *intrusion response* configuration may also be targeted by our optimisation-based framework. We note that one response to a suspected attack would be to reconfigure the IDS to provide more evidence regarding specific hypotheses

7.2.7 Dynamic reconfiguration

We have observed earlier that highly efficient determination of effective configurations is particularly important when repeated use of the framework is envisaged. It would be possible, for example, to repeatedly shift to a new configuration after a period of time as a means of preventing an adversary exploiting information learned about the current one during that period. This approach will raise issues, e.g. repeated change may impair a system manager's understanding of the operation of the system, but we believe the approach is worthy of investigation.

7.2.8 Explicable Machine Learning for Optimal Configurations

In Chapter 4, we argued that extracting insight from sets of reached configurations could be highly beneficial. In particular, we indicated that patterns of high performing solutions could be extracted. We argued that extracting such patterns was not easy and, indeed, an ML-based approach would be best suited to handle the inevitable complexity that would arise from network parameter variation (e.g., node wifi-range). This will likely be a challenging task for the Explicable Machine Learning community and a significant research topic in its own right.

7.2.9 Relaxing simplifying assumptions

For reasons of establishing proof of concept, we have made some simplifying choices. Most obviously, we adopted a very small set of very simple rules that could be deployed at a probe. These are simple representatives of detection capability. We could greatly extend the number and complexity of such rules. These rules could even be evolved as part of an advanced framework. Thus, we would not only evolve placement and rule choice, the very nature of those could be determined. A good deal of work has been done in this direction already (e.g. using genetic programming and grammatical evolution). Bring together placement, rule selection and rule nature under one optimisation-based framework will likely be computationally hugely intensive, but is a worthwhile goal.

Some simplifying assumptions can be easily relaxed. For example, we used single (global) thresholds for rule firing. Node-specific thresholds can readily be incorporated. Similarly, it is readily possible to use node-specific coverage levels.

7.3 Final remarks

We have explored the effectiveness and efficiency of an optimisation-based framework for IoT RPL network IDS configuration and have provided efficiency enhancements based on neural network models. Our approach has been shown to be beneficial and the future work above indicates that a great deal more can be done. We recommend the area of optimising IDS configurations to the IoT IDS research community.

Bibliography

[1] P. Perazzo, C. Vallati, G. Anastasi, and G. Dini, "Dio suppression attack against routing in the internet of things," *IEEE Communications Letters*, vol. 21, no. 11, pp. 2524–2527, 2017.

[2] J.-Y. Li, Z.-H. Zhan, and J. Zhang, "Evolutionary computation for expensive optimization: A survey," *Machine Intelligence Research*, vol. 19, no. 1, pp. 3–23, 2022.

[3] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[4] C. Sun, Y. Jin, J. Zeng, and Y. Yu, "A two-layer surrogate-assisted particle swarm optimization algorithm," *Soft computing*, vol. 19, no. 6, pp. 1461–1475, 2015.

[5] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," *Journal of microbiological methods*, vol. 43, no. 1, pp. 3–31, 2000.

[6] J. P. Anderson, "Computer security threat monitoring and surveillance," *Technical Report, James P. Anderson Company*, 1980.

[7] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A network security monitor," in *Proceedings. 1990 IEEE Computer Society Symposium on Research in Security and Privacy.* IEEE, 1990, pp. 296–304.

[8] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[9] T. Rao and E. Haq, "Security challenges facing iot layers and its protective measures," *International Journal of Computer Applications*, vol. 179, pp. 31–35, 03 2018.

[10] M. Centenaro, C. E. Costa, F. Granelli, C. Sacchi, and L. Vangelista, "A survey on technologies, standards and open challenges in satellite iot," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1693–1720, 2021.

[11] S. Sennan, R. Somula, A. K. Luhach, G. G. Deverajan, W. Alnumay, N. Jhanjhi, U. Ghosh, and P. Sharma, "Energy efficient optimal parent selection based routing protocol for internet of things using firefly optimization algorithm," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 8, p. e4171, 2021.

[12] Z. A. Khan and P. Herrmann, "Recent advancements in intrusion detection systems for the internet of things," *Security and Communication Networks*, vol. 2019, 2019.

[13] B. B. Gupta and M. Quamara, "An overview of internet of things (iot): Architectural aspects, challenges, and protocols," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 21, p. e4946, 2020.

[14] X.-S. Yang, S. Koziel, and L. Leifsson, "Computational optimization, modelling and simulation: Recent trends and challenges," *Procedia Computer Science*, vol. 18, pp. 855–860, 2013.

[15] J. Lewis, X. Zhong, and H. Rea, "A neural network approach to the robot inverse calibration problem," in *Second International Conference on Intelligent Systems Engineering, 1994*, 1994, pp. 342–347.

[16] J. Zhang, M. Ma, P. Wang, and X.-d. Sun, "Middleware for the internet of things: A survey on requirements, enabling technologies, and solutions," *Journal of Systems Architecture*, vol. 117, p. 102098, 2021.

[17] J. E. Tapiador and J. A. Clark, "The placement-configuration problem for intrusion detection nodes in wireless sensor networks," *Computers & Electrical Engineering*, vol. 39, no. 7, pp. 2306–2317, 2013.

[18] Y. S. Ong, P. Nair, A. Keane, and K. Wong, "Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems," in *Knowledge Incorporation in Evolutionary Computation.* Springer, 2005, pp. 307–331.

[19] C. Yan, M. Li, W. Liu, and M. Qi, "Improved adaptive genetic algorithm for the vehicle insurance fraud identification model based on a bp neural network," *Theoretical Computer Science*, vol. 817, pp. 12–23, 2020.

[20] P. Petersen and F. Voigtlaender, "Optimal approximation of piecewise smooth functions using deep relu neural networks," *Neural Networks*, vol. 108, pp. 296–330, 2018.

[21] S. Nižetić, P. Šolić, D. López-de-Ipiña González-de-Artaza, and L. Patrono, "Internet of things (iot): Opportunities, issues and challenges towards a smart and sustainable future," *Journal of Cleaner Production*, vol. 274, p. 122877, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095965262032922X

[22] Hampshire. (2020) Iot connections to reach 83 billion by 2024, driven by maturing industrial use cases. [Online]. Available: https://www.juniperresearch.com/press/iot-connections-to-reach-83-bn-by-2024

[23] J. Kandpal and A. Singh, "Opportunity and challenges for vlsi in iot application," in *5G Internet of Things and Changing Standards for Computing and Electronic Systems.* IGI Global, 2022, pp. 245–271.

[24] J. G. Monicka and C. Amuthadevi, "Smart automation, smart energy, and grid management challenges," *The Industrial Internet of Things (IIoT) Intelligent Analytics for Predictive Maintenance*, pp. 59–87, 2022.

[25] N. M. Karie, N. M. Sahri, and P. Haskell-Dowland, “Iot threat detection advances, challenges and future directions,” in *2020 workshop on emerging technologies for security in IoT (ETSecIoT)*. IEEE, 2020, pp. 22–29.

[26] P. M. Chanal and M. S. Kakkasageri, “Security and privacy in iot: a survey,” *Wireless Personal Communications*, vol. 115, no. 2, pp. 1667–1693, 2020.

[27] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, “Security of the internet of things: perspectives and challenges,” *Wireless Networks*, vol. 20, no. 8, pp. 2481–2501, 2014.

[28] I. Ali, S. Sabir, and Z. Ullah, “Internet of things security, device authentication and access control: a review,” *arXiv preprint arXiv:1901.07309*, 2019.

[29] K. Sha, W. Wei, T. A. Yang, Z. Wang, and W. Shi, “On security challenges and open issues in internet of things,” *Future Generation Comp. Syst.*, vol. 83, pp. 326–337, 2018.

[30] H. H. Addeen, Y. Xiao, J. Li, and M. Guizani, “A survey of cyber-physical attacks and detection methods in smart water distribution systems,” *IEEE Access*, vol. 9, pp. 99 905–99 921, 2021.

[31] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac, “Peek-a-boo: I see your smart home activities, even encrypted!” in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 207–218.

[32] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, “Internet of things security: A survey,” *Journal of Network and Computer Applications*, vol. 88, pp. 10–28, 2017.

[33] T. Watteyne, P. Thubert, and C. Bormann, “Rfc 8930: On forwarding 6lowpan fragments over a multi-hop ipv6 network,” *Internet Engineering Task Force*, 2020.

[34] Y. M. Haddad and H. H. Ali, “A dynamic approach for managing heterogeneous wireless networks in smart environments,” in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–7.

[35] C. Stergiou, K. E. Psannis, B.-G. Kim, and B. Gupta, “Secure integration of iot and cloud computing,” *Future Generation Computer Systems*, vol. 78, pp. 964–975, 2018.

[36] S. Rizvi, R. Orr, A. Cox, P. Ashokkumar, and M. R. Rizvi, “Identifying the attack surface for iot network,” *Internet of Things*, vol. 9, p. 100162, 2020.

[37] M. Alramadhan and K. Sha, “An overview of access control mechanisms for internet of things,” in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2017, pp. 1–6.

[38] W. Kassab and K. A. Darabkh, “A–z survey of internet of things: Architectures, protocols, applications, recent advances, future directions and recommendations,” *Journal of Network and Computer Applications*, vol. 163, p. 102663, 2020.

[39] T. Park, N. Abuzainab, and W. Saad, “Learning how to communicate in the internet of things: Finite resources and heterogeneity,” *IEEE Access*, vol. 4, pp. 7063–7073, 2016.

[40] B. Al-Duwairi, W. Al-Kahla, M. A. AlRefai, Y. Abedalqader, A. Rawash, and R. Fahmawi, "Siem-based detection and mitigation of iot-botnet ddos attacks," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 2, p. 2182, 2020.

[41] T. Gebremichael, L. P. I. Ledwaba, M. H. Eldefrawy, G. P. Hancke, N. Pereira, M. Gidlund, and J. Akerberg, "Security and privacy in the industrial internet of things: Current standards and future challenges," *IEEE Access*, vol. 8, pp. 152 351–152 366, 2020.

[42] M. Amiri-Zarandi, R. A. Dara, and E. Fraser, "A survey of machine learning-based solutions to protect privacy in the internet of things," *Computers & Security*, vol. 96, p. 101921, 2020.

[43] P. Shi, H. Wang, S. Yang, C. Chen, and W. Yang, "Blockchain-based trusted data sharing among trusted stakeholders in iot," *Software: practice and experience*, vol. 51, no. 10, pp. 2051–2064, 2021.

[44] O. Kayode and A. S. Tosun, "Analysis of iot traffic using http proxy," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[45] H. Kharrufa, H. A. Al-Kashoash, and A. H. Kemp, "Rpl-based routing protocols in iot applications: a review," *IEEE Sensors Journal*, vol. 19, no. 15, pp. 5952–5967, 2019.

[46] T. Winter, P. Thubert, A. Brandt, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, R. K. Alexander *et al.*, "Rpl: Ipv6 routing protocol for low-power and lossy networks." *rfc*, vol. 6550, pp. 1–157, 2012.

[47] Z. A. Almusaylim, A. Alhumam, and N. Jhanjhi, "Proposing a secure rpl based internet of things routing protocol: A review," *Ad Hoc Networks*, vol. 101, p. 102096, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870519308388

[48] A. Althubaity, T. Gong, K.-K. Raymond, M. Nixon, R. Ammar, and S. Han, "Specification-based distributed detection of rank-related attacks in rpl-based resource-constrained real-time wireless networks," in *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, vol. 1. IEEE, 2020, pp. 168–175.

[49] F. Medjek, D. Tandjaoui, M. R. Abdmeziem, and N. Djedjig, "Analytical evaluation of the impacts of sybil attacks against rpl under mobility," in *2015 12th International Symposium on Programming and Systems (ISPS)*. IEEE, 2015, pp. 1–9.

[50] R. Stenhuis, "Rpl attack analysis: Evaluation of a cryptography-based sybil defence in ieee 802.15. 4," Delft University of Technology, Tech. Rep., Jul. 2021.

[51] A. B. Ordu, M. Bayar, and B. Örs, "Rpl authenticated mode evaluation: Authenticated key exchange and network behavioral," in *2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2022, pp. 167–173.

[52] D. Airehrour, J. A. Gutierrez, and S. K. Ray, "Sectrust-rpl: A secure trust-aware rpl routing protocol for internet of things," *Future Generation Computer Systems*, vol. 93, pp. 860–876, 2019.

[53] O. Can and O. K. Sahingoz, “A survey of intrusion detection systems in wireless sensor networks,” in *2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*. IEEE, 2015, pp. 1–6.

[54] M. Alsharif and D. B. Rawat, “Study of machine learning for cloud assisted iot security as a service,” *Sensors*, vol. 21, no. 4, p. 1034, 2021.

[55] A. M. Pasikhani, J. A. Clark, P. Gope, and A. Alshahrani, “Intrusion detection systems in rpl-based 6lowpan: a systematic literature review,” *IEEE Sensors Journal*, 2021.

[56] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, “Detection of sinkhole attacks for supporting secure routing on 6lowpan for internet of things,” in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 606–611.

[57] P. Kasinathan, G. Costamagna, H. Khaleel, C. Pastrone, and M. A. Spirito, “An ids framework for internet of things empowered by 6lowpan,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 1337–1340.

[58] A. M. da Silva Cardoso, R. F. Lopes, A. S. Teles, and F. B. V. Magalhães, “Real-time ddos detection based on complex event processing for iot,” in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2018, pp. 273–274.

[59] J. P. Amaral, L. M. Oliveira, J. J. Rodrigues, G. Han, and L. Shu, “Policy and network-based intrusion detection system for ipv6-enabled wireless sensor networks,” in *2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, pp. 1796–1801.

[60] A. Le, J. Loo, K. Chai, and M. Aiash, “A specification-based ids for detecting attacks on rpl-based network topology,” *Information*, vol. 7, no. 2, p. 25, 2016.

[61] A. Amouri, V. T. Alaparthy, and S. D. Morgera, “Cross layer-based intrusion detection based on network behavior for iot,” in *2018 IEEE 19th Wireless and Microwave Technology Conference (WAMICON)*. IEEE, 2018, pp. 1–4.

[62] T. Sherasiya and H. Upadhyay, “Intrusion detection system for internet of things,” *Int. J. Adv. Res. Innov. Ideas Educ.(IJARIIE)*, vol. 2, pp. 2244–2259, 2016.

[63] P. Loulianou, V. Vasilakia, I. Moscholios, and M. Logothetis, “A signature-based intrusion detection system for the internet of things,” *Information and Communication Technology Form*, 2018.

[64] N. K. Thanigaivelan, E. Nigussie, R. K. Kanth, S. Virtanen, and J. Isoaho, “Distributed internal anomaly detection system for internet-of-things,” in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2016, pp. 319–320.

[65] D. Midi, A. Rullo, A. Mudgerikar, and E. Bertino, “Kalis—a system for knowledge-driven adaptable intrusion detection for the internet of things,” in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS).* IEEE, 2017, pp. 656–666.

[66] M. Roesch *et al.*, “Snort: Lightweight intrusion detection for networks.” in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.

[67] S. A. Shaikh, H. Chivers, P. Nobles, J. A. Clark, and H. Chen, “A deployment value model for intrusion detection sensors,” in *International Conference on Information Security and Assurance.* Springer, 2009, pp. 250–259.

[68] B. Farzaneh, M. A. Montazeri, and S. Jamali, “An anomaly-based ids for detecting attacks in rpl-based internet of things,” in *2019 5th International Conference on Web Research (ICWR).* IEEE, 2019, pp. 61–66.

[69] M. Yadollahzadeh Tabari and Z. Mataji, “Detecting sinkhole attack in rpl-based internet of things routing protocol,” *Journal of AI and Data Mining*, vol. 9, no. 1, pp. 73–85, 2021.

[70] P. Pongle and G. Chavan, “Real time intrusion and wormhole attack detection in internet of things,” *International Journal of Computer Applications*, vol. 121, no. 9, 2015.

[71] D. Shreenivas, S. Raza, and T. Voigt, “Intrusion detection in the rpl-connected 6lowpan networks,” in *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security.* ACM, 2017, pp. 31–38.

[72] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder, “Using the rpl protocol for supporting passive monitoring in the internet of things,” in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium.* IEEE, 2016, pp. 366–374.

[73] S. Sonavane, “Design and implementation of rssi based intrusion detection system for rpl based iot network,” *Int. J. Comput. Sci. Netw. Secur.*, vol. 19, no. 12, pp. 1–9, 2020.

[74] C. Pu, “Sybil attack in rpl-based internet of things: analysis and defenses,” *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4937–4949, 2020.

[75] A. Verma and V. Ranga, “Mitigation of dis flooding attacks in rpl-based 6lowpan networks,” *Transactions on emerging telecommunications technologies*, vol. 31, no. 2, p. e3802, 2020.

[76] D. B. Gothawal and S. Nagaraj, “Anomaly-based intrusion detection system in rpl by applying stochastic and evolutionary game models over iot environment,” *Wireless Personal Communications*, vol. 110, no. 3, pp. 1323–1344, 2020.

[77] B. Ghaleb, A. Al-Dubai, E. Ekonomou, M. Qasem, I. Romdhani, and L. Mackenzie, “Addressing the dao insider attack in rpl’s internet of things networks,” *IEEE Communications Letters*, vol. 23, no. 1, pp. 68–71, 2018.

[78] A. Verma and V. Ranga, "Elnids: Ensemble learning based network intrusion detection system for rpl based internet of things," in *2019 4th International conference on Internet of Things: Smart innovation and usages (IoT-SIU)*. IEEE, 2019, pp. 1–6.

[79] S. Choudhary and N. Kesswani, "Cluster-based intrusion detection method for internet of things," in *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*. IEEE, 2019, pp. 1–8.

[80] E. Aydogan, S. Yilmaz, S. Sen, I. Butun, S. Forsström, and M. Gidlund, "A central intrusion detection system for rpl-based industrial internet of things," in *2019 15th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 2019, pp. 1–5.

[81] J. Li, Z. Zhao, R. Li, and H. Zhang, "Ai-based two-stage intrusion detection for software defined iot networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2093–2102, 2018.

[82] J. Foley, N. Moradpoor, and H. Ochenyi, "Employing a machine learning approach to detect combined internet of things attacks against two objective functions using a novel dataset," *Security and Communication Networks*, vol. 2020, 2020.

[83] S. A. Shaikh, H. Chivers, P. Nobles, J. A. Clark, and H. Chen, "Characterising intrusion detection sensors," *Network Security*, vol. 2008, no. 9, pp. 10–12, 2008.

[84] A. Stetsko, T. Smolka, V. Matyáš, and M. Stehlík, "Improving intrusion detection systems for wireless sensor networks," in *International Conference on Applied Cryptography and Network Security*. Springer, 2014, pp. 343–360.

[85] H. Jh, "Adaptation in natural and artificial systems," *Ann Arbor*, 1975.

[86] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091–8126, 2021.

[87] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[88] K. Deb, "An introduction to genetic algorithms," *Sadhana*, vol. 24, no. 4, pp. 293–315, 1999.

[89] S. A. Haroun, B. Jamal *et al.*, "A performance comparison of ga and aco applied to tsp," *International Journal of Computer Applications*, vol. 117, no. 20, 2015.

[90] H. Chen, J. A. Clark, S. A. Shaikh, H. Chivers, and P. Nobles, "Optimising ids sensor placement," in *2010 International Conference on Availability, Reliability and Security*. IEEE, 2010, pp. 315–320.

[91] S. K. Gupta, P. Kuila, and P. K. Jana, "Genetic algorithm approach for k-coverage and m-connected node placement in target based wireless sensor networks," *Computers & Electrical Engineering*, vol. 56, pp. 544–556, 2016.

[92] Y. Fang and J. Li, "A review of tournament selection in genetic programming," in *International Symposium on Intelligence Computation and Applications.* Springer, 2010, pp. 181–192.

[93] G. Acampora, A. Chiatto, and A. Vitiello, "Training variational quantum circuits through genetic algorithms," in *2022 IEEE Congress on Evolutionary Computation (CEC).* IEEE, 2022, pp. 1–8.

[94] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, and V. S. Prasath, "Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach," *Information*, vol. 10, no. 12, p. 390, 2019.

[95] F. Anjum, D. Subhadrabandhu, S. Sarkar, and R. Shetty, "On optimal placement of intrusion detection modules in sensor networks," in *First International Conference on Broadband Networks.* IEEE, 2004, pp. 690–699.

[96] M. Rolando, M. Rossi, N. Sanarico, and D. Mandrioli, "A formal approach to sensor placement and configuration in a network intrusion detection system," in *Proceedings of the 2006 international workshop on Software engineering for secure systems.* ACM, 2006, pp. 65–71.

[97] S. Noel and S. Jajodia, "Attack graphs for sensor placement, alert prioritization, and attack response," in *Cyberspace Research Workshop*, 2007, pp. 1–8.

[98] B. Mishra and I. Smirnova, "Optimal configuration of intrusion detection systems," *Information Technology and Management*, pp. 1–14, 2021.

[99] M. Stehlik, A. Saleh, A. Stetsko, and V. Matyas, "Multi-objective optimization of intrusion detection systems for wireless sensor networks," in *ECAL 2013: The Twelfth European Conference on Artificial Life.* MIT Press, 2013, pp. 569–576.

[100] A. Hassanzadeh and R. Stoleru, "Towards optimal monitoring in cooperative ids for resource constrained wireless networks," in *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN).* IEEE, 2011, pp. 1–8.

[101] Hassanzadeh, Amin and Stoleru, Radu, "On the optimality of cooperative intrusion detection for resource constrained wireless networks," *Computers & Security*, vol. 34, pp. 16–35, 2013.

[102] M. Krzysztoń and M. Marks, "Simulation of watchdog placement for cooperative anomaly detection in bluetooth mesh intrusion detection system," *Simulation Modelling Practice and Theory*, vol. 101, p. 102041, 2020.

[103] U. Ghugar, J. Pradhan, and M. Biswal, "A novel intrusion detection system for detecting black hole attacks in wireless sensor network using aodv protocol," *IJCSN-International Journal of Computer Science and Network*, vol. 5, no. 4, 2016.

[104] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft computing*, vol. 9, no. 1, pp. 3–12, 2005.

[105] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, "Data-driven evolutionary optimization: An overview and case studies," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 442–458, 2018.

[106] S. Liang and R. Srikant, "Why deep neural networks for function approximation?" *arXiv preprint arXiv:1610.04161*, 2016.

[107] S. Yang, T. Ting, K. L. Man, and S.-U. Guan, "Investigation of neural networks for function approximation," *Procedia Computer Science*, vol. 17, pp. 586–594, 2013.

[108] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[109] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[110] O. Iova, P. Picco, T. Istomin, and C. Kiraly, "Rpl: The routing standard for the internet of things... or is it?" *IEEE Communications Magazine*, vol. 54, no. 12, pp. 16–22, 2016.

[111] A. M. Pasikhani, J. A. Clark, P. Gope, and A. Alshahrani, "Intrusion detection systems in rpl-based 6lowpan: A systematic literature review," *IEEE Sensors Journal*, 2021.

[112] A. Ferdowsi and W. Saad, "Generative adversarial networks for distributed intrusion detection in the internet of things," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[113] T. M. Behera, S. K. Mohapatra, U. C. Samal, M. S. Khan, M. Daneshmand, and A. H. Gandomi, "I-sep: An improved routing protocol for heterogeneous wsn for iot-based environmental monitoring," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 710–717, 2020.

[114] A. Ghafouri, W. Abbas, A. Laszka, Y. Vorobeychik, and X. Koutsoukos, "Optimal thresholds for anomaly-based intrusion detection in dynamical environments," in *Decision and Game Theory for Security*, Q. Zhu, T. Alpcan, E. Panaousis, M. Tambe, and W. Casey, Eds. Cham: Springer International Publishing, 2016, pp. 415–434.

[115] M. Mahyoub, A. S. H. Mahmoud, M. Abu-Amara, and T. R. Sheltami, "An efficient rpl-based mechanism for node-to-node communications in iot," *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7152–7169, 2020.

[116] M. Jamshidi, S. S. A. Poor, A. Arghavani, M. Esnaashari, A. A. Shaltooki, and M. R. Meybodi, "A simple, lightweight, and precise algorithm to defend against replica node attacks in mobile wireless networks using neighboring information," *Ad Hoc Networks*, vol. 100, p. 102081, 2020.

[117] M. Alzubaidi, M. Anbar, and S. M. Hanshi, "Neighbor-passive monitoring technique for detecting sinkhole attacks in rpl networks," in *Proceedings of the 2017 International Conference on Computer Science and Artificial Intelligence*, 2017, pp. 173–182.

[118] M. Amjad, M. K. Afzal, T. Umer, and B.-S. Kim, "Qos-aware and heterogeneously clustered routing protocol for wireless sensor networks," *IEEE Access*, vol. 5, pp. 10 250–10 262, 2017.

[119] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 992–1007, 2006, special Issue - Genetic Algorithms and Reliability. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0951832005002012

[120] Tetcos: NetSim - Network Simulation Software, "Netsim," 2021. [Online]. Available: https://www.tetcos.com/

[121] A. Rasch, M. Haidl, and S. Gorlatch, "Atf: A generic auto-tuning framework," in *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2017, pp. 64–71.

[122] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou, "Metamodel—assisted evolution strategies," in *International Conference on parallel problem solving from nature*. Springer, 2002, pp. 361–370.

[123] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[124] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, "Kerastuner," https://github.com/keras-team/keras-tuner, 2019.

[125] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[126] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine learning with oversampling and undersampling techniques: overview study and experimental results," in *2020 11th international conference on information and communication systems (ICICS)*. IEEE, 2020, pp. 243–248.

[127] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[128] P. Branco, L. Torgo, and R. P. Ribeiro, "Smogn: a pre-processing approach for imbalanced regression," in *First international workshop on learning with imbalanced domains: Theory and applications*. PMLR, 2017, pp. 36–50.

[129] G. E. Box and D. R. Cox, "An analysis of transformations," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 26, no. 2, pp. 211–243, 1964.

[130] A. Le, J. Loo, K. K. Chai, and M. Aiash, "A specification-based ids for detecting attacks on rpl-based network topology," *Information*, vol. 7, no. 2, 2016. [Online]. Available: https://www.mdpi.com/2078-2489/7/2/25

[131] A. Bajaj and O. P. Sangwan, "A systematic literature review of test case prioritization using genetic algorithms," *IEEE Access*, vol. 7, pp. 126 355–126 375, 2019.

[132] S. Pandey and S. Varma, “A range based localization system in multihop wireless sensor networks: A distributed cooperative approach,” *Wireless Personal Communications*, vol. 86, no. 2, pp. 615–634, 2016.

[133] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279.

[134] X. Ying, “An overview of overfitting and its solutions,” *Journal of Physics: Conference Series*, vol. 1168, p. 022022, feb 2019. [Online]. Available: https://doi.org/10.1088/1742-6596/1168/2/022022

[135] S. Shamshirband, D. Petković, H. Javidnia, and A. Gani, “Sensor data fusion by support vector regression methodology—a comparative study,” *IEEE Sensors Journal*, vol. 15, no. 2, pp. 850–854, 2014.

[136] O. Gaddour, A. Koubäa, R. Rangarajan, O. Cheikhrouhou, E. Tovar, and M. Abid, “Co-rpl: Rpl routing for mobile low power wireless sensor networks using corona mechanism,” in *Proceedings of the 9th IEEE international symposium on industrial embedded systems (SIES 2014)*. IEEE, 2014, pp. 200–209.