Identifying and Detecting Attacks in Industrial Control Systems

Nilufer Tuptuk

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

UCL DEPARTMENT OF COMPUTER SCIENCE

UCL DEPARTMENT OF SECURITY AND CRIME SCIENCE

University College London

May 29, 2019

I, Nilufer Tuptuk, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

The integrity of industrial control systems (ICS) found in utilities, oil and natural gas pipelines, manufacturing plants and transportation is critical to national well-being and security. Such systems depend on hundreds of field devices to manage and monitor a physical process. Previously, these devices were specific to ICS but they are now being replaced by general purpose computing technologies and, increasingly, these are being augmented with Internet of Things (IoT) nodes.

Whilst there are benefits to this approach in terms of cost and flexibility, it has attracted a wider community of adversaries. These include those with significant domain knowledge, such as those responsible for attacks on Iran's Nuclear Facilities, a Steel Mill in Germany, and Ukraine's power grid; however, non-specialist attackers are becoming increasingly interested in the physical damage it is possible to cause. At the same time, the approach increases the number and range of vulnerabilities to which ICS are subject; regrettably, conventional techniques for analysing such a large attack space are inadequate, a cause of major national concern.

In this thesis we introduce a generalisable approach based on evolutionary multiobjective algorithms to assist in identifying vulnerabilities in complex heterogeneous ICS systems. This is both challenging and an area that is currently lacking research. Our approach has been to review the security of currently deployed ICS systems, and then to make use of an internationally recognised ICS simulation testbed for experiments, assuming that the attacking community largely lack specific ICS knowledge. Using the simulator, we identified vulnerabilities in individual components and then made use of these to generate attacks. A defence against these attacks in the form of novel intrusion detection systems were developed, based on

a range of machine learning models. Finally, this was further subject to attacks created using the evolutionary multiobjective algorithms, demonstrating, for the first time, the feasibility of creating sophisticated attacks against a well-protected adversary using automated mechanisms.

Impact Statement

The objective of the research described in this thesis is to develop an approach that can be used to identify vulnerabilities in Industrial Control Systems (ICS). ICS are used to monitor and control many of the national critical services, including energy production, manufacturing plants, chemical processing, financial services, transportation and healthcare, to a name but a few. Any failures or disruptions can have severe consequences, from economic damage and lost production, through injury and loss of life, to catastrophic nation-wide effects. The security of these systems is therefore important to national security as well as individual manufacturing enterprises. As a result, research in this area is of significant interest to policy makers, industry and academia.

Given the novelty of Internet-facing ICS to the attacker communities, the potential variety of attacks is extensive and unknown. The only way we can begin to understand our exposure to the risk of attack is to conduct practical research studies in which systems and countermeasures are actively attacked in advance of deployment. Recognising that conducting experiments on real systems is not safe and is often not feasible, this research offers an alternative: to create realistic ICS testbeds, and carry out attacks diagnosed to identify vulnerabilities proactively instead of waiting for the adversaries to find these vulnerabilities. Our methodology is a new approach to the design of better security. It can be implemented by those that do not have special security knowledge needed to test their systems for vulnerabilities, in a more conventional manner to help them in making security related decisions.

This research has further benefits for academia. The research carried out ap-

plies to several academic domains: control engineering, information security and evolutionary computation. Control engineers can use the insight gained from this work to analyse the implications of security on process control, and to design security-aware control algorithms. The detection models we built will interest the information security, machine learning and the fault detection and diagnosis communities. The training dataset we have generated in this thesis contains a rich set of attacks; this will be made opensource and will provide a new resource to the research community in this area who are currently reliant on a limited set of examples.

We designed a methodology for securing ICS using evolutionary multiobjective optimisation algorithms, and applied this to a domain that is more realistic than has been. Existing studies that make use of adversarial machine learning and evolutionary computation tend to focus on simplified examples.

ICS security is currently on the agenda of many governmental bodies, standards bodies and business. This research will be beneficial in understanding some of the security threats we are facing, both as the community of attackers changes and as ICS become increasingly Internet facing. Finally, the potential of our approach to handle complexity makes it ideal for application to the IoT systems that underpin Industry 4.0 since these have more complex dynamics and a much larger attack surface.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor, Prof. Stephen Hailes for his constant guidance, patience, motivation and support throughout the completion of this thesis. Thank you for always being available, you were always very generous, kind and helpful.

Thank you to my second supervisor Prof. David Pym. Thank you to Dr. Bill Langdon and Dr. Hector Menendez for showing interest in my work, and helping me to understand the field of evolutionary computation.

I am grateful to Prof. Richard Wortley for his valuable support throughout the years, and giving me the opportunity to do this PhD as part of the UCL Security Science Doctoral Training Centre (SECReT). I would like to also say thank you to administrative staff for their support, and my fellow PhD students at the Department of Security and Crime Science for all the good times and laughs.

Thank you to staff and students at the Department of Computer Science for creating a positive and enjoyable work environment.

To all my friends: thank you for your friendship, support and patience throughout the years, with special thanks to Rae and Veronika, for always cheering me up and listening to me.

I am thankful to my parents, my sisters, my three year and a half old niece Aliya (for reminding me that “a good rabbit never gives up”), and my extended family for their love, support and encouragement, without them none of this would have been possible.

Finally, I would like to express my gratitude for the generous funding awarded by the Engineering and Physical Sciences Research Council (EPSRC). The research

presented here was made possible through the grant provided via the EPSRC Security Science Doctoral Training Centre with reference number EP/G037264/1. The final year of this work was supported by the PETRAS – EPSRC IoT Research Hub.

Contents

List of Figures

List of Tables

LIST OF ABBREVIATIONS

AdaBoost Adaptive Boosting

ANN Artificial Neural Network

CART Classification And Regression Trees algorithm

COTS Commercial off-the-shelf

DCS Distributed Control Systems

DoS Denial of Service

EMO Evolutionary Multiobjective Optimisation

EMOA Evolutionary Multiobjective Optimisation Algorithms

EWMA Exponentially Weighted Moving Average

FNN Feedforward Neural Network

GA Genetic Algorithm

GP Genetic Programming

GRU Gated Recurrent Units

ICS Industrial Control Systems

IDS Intrusion Detection Systems

LSTM Long Short-Term Memory

MOO Multiobjective Optimisation

MSE Mean Square Error

MitM MitM Man-in-the-middle

NCS Networked Control System

NSGA-II Non-Dominated Sorting Genetic Algorithm II

PLC Programming Logic Controller

RAT Remote Access Trojan

RBF Radial Basis Function

RNN Recurrent Neural Network

SCADA System Collection and Data Acquisition

SPEA2 Strength Pareto Evolutionary Algorithm 2

SVM Support Vector Machine

TE Tennessee Eastman Control Process

XMEAS Process Variable Measurements Vector

XMV Manipulated Variable Vector

Chapter 1

Introduction

An Industrial Control System (ICS) is an instance of a more general class of cyber-physical system (CPS), and the term is used to describe the command and control systems that are found at the core of the national critical infrastructure services. The industry covered by these areas includes: gas; electricity; oil; water treatment; telecommunication; transportation; process manufacturing (chemicals, pharmaceuticals, paper, food, beverages and other batched-based manufacturers); and discrete manufacturing (automobiles, ships, computers and many other durable goods).

The security of ICS is an important concern in modern society as national security, economic prosperity and public health and safety rely on the services of these systems. In the past, security of ICS was achieved simply through isolation and control of physical access. However, for reasons of economics and convenience, it is increasingly becoming the case that the adoption of networks as a complement to bus technologies, the increasing use of commercial-off-the-shelf (COTS) components, and the deployment of wireless systems are becoming a core part of modern factories. Factory and plant networks are often connected to the wider corporate network to increase efficiency of production and reduce the risk of the downtime that costs manufacturing 33% of its profits per year [1]. Similarly, to extend network infrastructure to remote field areas, increase sensing capacity, handle mobility and reduce installation costs, there is an increase in the member of wireless networks being deployed. These approaches are leaving ICS networks vulnerable. To give a sense of the scale of this vulnerability, according to outputs from Project SHINE

there were, in 2012, in excess of 500,000 Internet-accessible devices that could be loosely classified as control system devices [2]. The consequences are substantial: by making ICS easier to access, the number and the geographic spread of potential adversaries increases massively, and the risks to the attacker are reduced both because it is hard to establish their identity and because they may anyway be in a different legal jurisdiction.

The number of motivated and highly skilled adversaries carrying out complex attacks targeting critical infrastructures is also on the increase. As the evidence from successful attacks [3] [4] [5] [6] [7] shows, they can cause catastrophic consequences. The potential outcome of a successful attack on a critical service ranges from injuries and fatalities, through serious damage to the environment, to catastrophic nation-wide economic loss due to production losses or degradation of products and services. Shutting down or denying access to these systems (for example those operating the production and distribution of utilities such as electricity, water and gas), even for a short time, may cause significant harm to people and erode the public's confidence, leading to a general feeling of insecurity.

Despite the technological advances in ICS, understanding the vulnerabilities of systems that have a large attack space, and the consequences when these vulnerabilities are exploited, remains a major concern. In this thesis, we develop an approach that can assist in determining the vulnerabilities of these complex heterogeneous ICS systems at the process level. Our aim is to produce a novel reactive security tool that can improve security against ICS attacks.

1.1 Motivation

Industrial control systems are complex systems integrating computing, communication protocols and controls. Figure 1.1 shows the elements of a simple closed-loop system, as used in the operation of most automated industrial processes. The desired output of the system is given as a reference signal (setpoint). The sensor measures the value of the process, and the controller compares the setpoint variable to the sensor reading, and adjusts the system based on the difference between them (error).

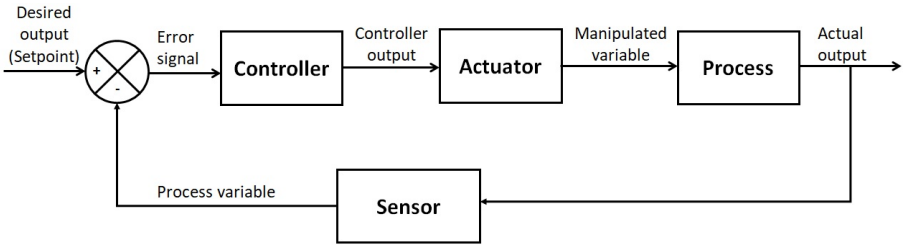


Figure 1.1: A typical closed loop control

Adjustment is done according to some control algorithm, and appropriate signals are sent to the actuators in an attempt to reduce the error. Actuators drive the plant by modulating the process according to the input received from the controller. This kind of control has ability to maintain a physical process at the desired value in the presence of external disturbances [8]. The entirety of this process is monitored by the operators and engineers using special computers and a group of diagnostic and maintenance utilities to detect, prevent and respond to faults and any other unusual behaviour [9]. An adversary who is trying to cause damage will carry out attacks that aim to push the plant away from the setpoint by manipulating the signals in some part of the control loop.

The signals from sensor to controller (process variable measurements) and controller to actuator (manipulated variables) are no longer sent using traditional point-to-point ICS communication in which a wire connects the controller to the sensors and actuators. Instead, there has been a move towards networked control systems (NCS), in which the communication between spatially distributed controllers, sensors and actuators is carried over communication networks [10]. This means the basic feedback control system illustrated in Figure 1.1 is increasingly being closed using a shared network.

With technological advances, factory and plant networks are becoming highly connected over multiple layers, creating opportunities for a range of adversaries aiming to reach the process level. Figure 1.2 illustrates the network levels that are likely to be found in these environments [11]. Level 4, the corporate network, is the level at which business decisions are made, and in which the regular corporate systems (enterprise desktops and servers) operate. At Level 3, the operation

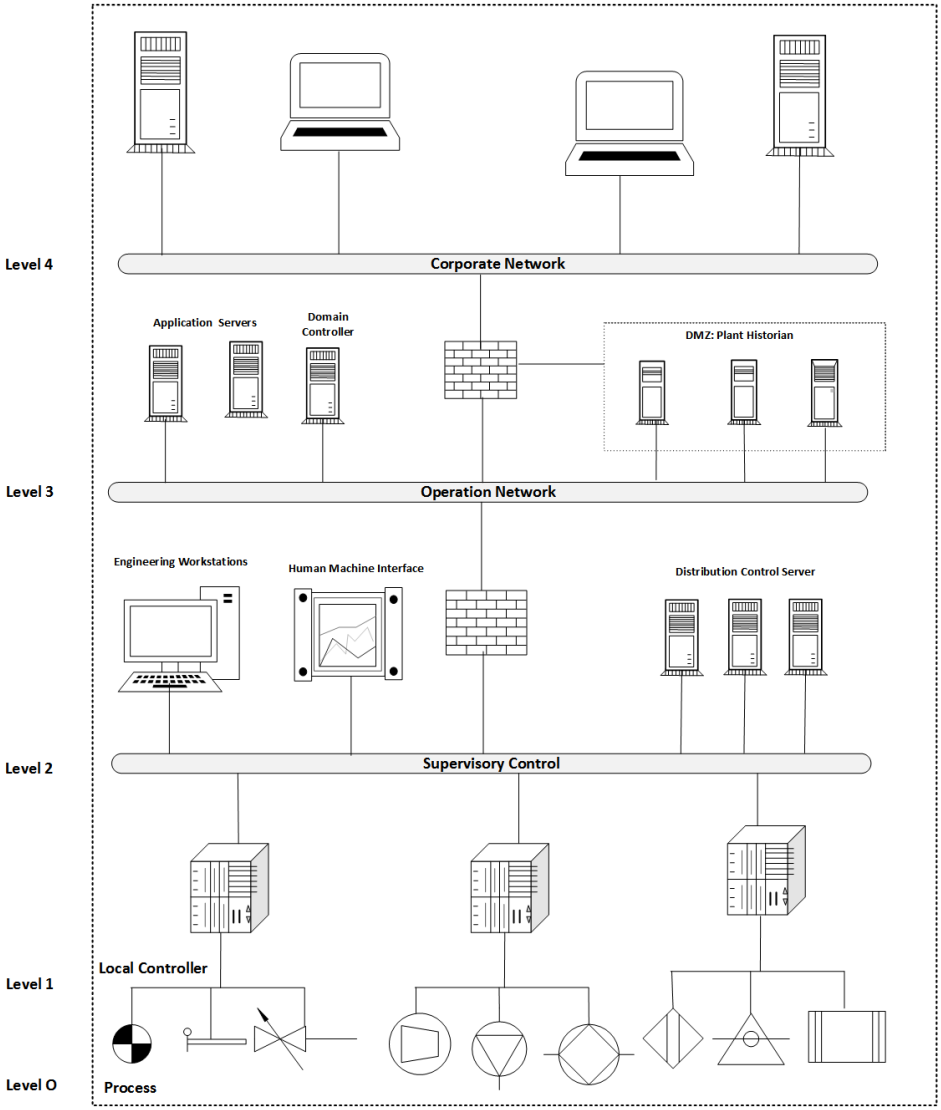


Figure 1.2: Industrial network architecture layers

network, operational management systems such as domain controllers, data collection servers (historians) and application servers are found. Level 2, supervisory control, consists of devices that monitor and control the process at the lower levels. Typically, these consist of supervisory interfaces for the operators, engineering workstations, and distributed control server monitoring and controlling various parts of the production. At Level 1, controllers monitor and control a set of devices according to decisions that come from the supervisory system. They receive inputs from process equipment (e.g. field devices) such as sensors, and send output signals to the other devices (actuators). Level 0 is where the actual process takes

place, containing the sensors and actuators connected via a fieldbus network. It is worth dismissing one common misconception here: at the moment, these networks are typically not protected from unauthorised access through strong access control mechanisms. If security exists at all, it is usually minimal, with the use of firewalls to protect the input and output traffic between the networks, as illustrated in Figure 1.2. These vulnerabilities open doors to a range of adversaries with different motivations. However, to be able to cause physical damage against the process, the adversaries will need to manipulate the process directly by altering the parameters set at Level 0. The focus of this thesis is to analyse the vulnerabilities at this level, the sensor-control-actuator level.

1.1.1 Attacks against Industrial Control Systems

[*Some of the text in this section has been published in [12]*]. At present, there have been rather few publicly reported attacks against industrial control systems. The attack against Ukraine's power grid in December 2015 [13] caused several blackouts, causing 225,000 customers to lose power across Ukraine. The attack in December 2014, against a steel factory in Germany, caused massive physical, damage to the system by manipulating individual control components, thereby bringing the blast furnace under the control of the attackers. The skill sets required to carry out this attack were not only in the field of information security, but extended to control systems and production processes [14]. In 2014, a Remote Access Trojan (RAT) called Havex/Dragonfly was used to compromise industrial control systems including Programmable Logic Controllers (PLC), Distributed Control Systems (DCS) and System Collection and Data Acquisition (SCADA) systems used within the energy sector [15] across the globe. In 2011, a sophisticated instance of a RAT, known as Duqu [7], infected control systems in Europe, Asia and North Africa. Duqu's payload modules contained remote access capabilities that were used to connect to a command and control (C&C) server. Stuxnet [3], first reported in 2010, is believed to be the first worm that was designed with the sole aim of causing physical damage. It specifically targeted Iran's uranium enrichment facility at the Natanz enrichment plant. Researchers estimated Stuxnet may have destroyed about 1,000

(10%) of the centrifuges installed at the time of the attack [3].

These attacks demonstrated that specialist knowledge is required to launch effective targeted attacks and, in the past, these systems might have been targeted only by adversaries with this significant domain knowledge. At present, we might still be protected only by the difficulties inherent in the process of launching an attack on a specialised system; however, this is changing.

Some predict that the number of "things" connected to the Internet, will reach a number between 20 and 50 billion by 2020 [16] [1]. The integration of Internet of Things (IoT) [17] technologies into industrial systems is widely seen as the next logical step in the move towards Industry 4.0 and building smarter factories. This vision promises to enhance productivity by increasing efficiency, providing better management of processes, and being more predictable with the use of technologies and cloud analytics. The ICS sectors that will be most impacted by this growth are utilities, manufacturing and government. Specifically, the adoption of IoT will be a critical element of future smart factories and plants. For ICS, this will mean replacing the existing integrated infrastructure with a decentralised autonomous model in which devices will become active participants in the IoT. These devices will act as autonomous intelligent agents that are not exclusively under the control of a central controller. Decentralised decision making facilitated by wireless communication, is key to this vision. Such an open environment, with increased connectivity and data sharing, promises unprecedented convenience and long-term economic potential.

However, it is also prone to a wide range of both passive and active security attacks ranging from conventional eavesdropping and denial of service (DoS) attacks to man-in-the-middle (MitM) attacks that subtly alter the quality or consistency of the end product. Furthermore, adapting COTS technologies such as workstations running well-known operating systems (Microsoft Windows, Linux) has reduced installation costs and provided greater interconnectivity; however, these also inherit the vulnerabilities of such products, and so provide attackers with the opportunity to use existing exploits and well developed scanning methods amongst other things to attack these systems. As a result, it is no longer possible to assume that those

attacking these systems require special skills or capabilities, such as might be found at the level of foreign intelligence services and nation states. Instead, these systems will attract a range of adversaries with non-specialist capabilities, including criminals and hackers. Furthermore, with an increased number of connections in ICS, adversaries can organise and acquire the necessary skills and intelligence to get into these systems remotely.

1.1.2 Security of Industrial Control Systems

The security of ICS is an area that has received relatively little attention and scrutiny by the security community. This is due to lack of testbeds, datasets and domain knowledge. By and large, the focus of the existing security research relates to the vulnerabilities of the industrial communication networks and protocols. However, it is possible that adversaries can evade detection at the network level and, since there is an increasing use of COTS-based technology, assuming the network is secure is likely to be a mistake. Once an adversary reaches the process level, they can carry out attacks with the intention of causing damage to the process, equipment or even in violating regulatory compliance requirements [18].

A variety of attack types can be perpetrated against the ICS components involved directly in the process: these involve the modifications of process variable measurements (sensor measurements) or manipulated variables (values going from controller to actuators), or manipulation of the control algorithm (e.g. set points). Real-world ICS systems are complex systems with a large number of devices and non-linear dependencies. Although there are some studies that have looked at single attack instances, we are not aware of any adversarial studies exploring the control processes in the presence of combinations of attack instances, let alone doing this in an automated manner. This is a problem that can be solved by searching; however, conventional search methods are not suitable for identifying attacks in this space. Evolutionary algorithms are a class of stochastic population-based optimisation techniques that take their inspiration from neo-Darwinian evolution. Instead of working from a single solution, these algorithms start from a population of potential solutions (often generated randomly), and evolve towards better solutions using op-

erators inspired by biological evolution and genetics. They have been successfully applied to hard optimisation problems, and their applications can be found in engineering, manufacturing, transportation, robotics, medicine, software engineering and security. Motivated by the existing work in this area, we decided to investigate evolutionary algorithms as a means of analysing ICS security. We believe this approach can be utilised by the control community to design security-aware control algorithms, since it can be applied even by those without a deep knowledge of security.

Intrusion detection systems play an important role in early detection of attacks in industrial control. There is a considerable general literature on Intrusion detection systems (IDS) as applied to networked systems. The focus of most of the earlier studies in this area has been on the network level: intrusion detection systems looking for abnormal behaviour or patterns. The detection of attacks that are carried out to disrupt a physical process is only just starting to receive more attention; however, analogous situations have been studied more thoroughly by the control community in the context of fault detection and fault diagnosis. The weaknesses of many of the proposed approaches is that they assume that faults are known prior to training; this is a strong assumption because obtaining anomalous behaviour (attacks or faults), as obtaining anomalous samples may try to avoid detection; as a result, their behaviour is often different to random faults.

Many machine learning techniques, both supervised and unsupervised, have been successfully applied to detection of abnormal behaviour. Given the vast amount of data produced by ICS systems, there is an increasing number of studies proposing deep learning neural networks as a means of detecting anomalous behaviour. However these techniques are subject to attack: adversaries may seek to compromise what is learned and so evade detection. Such attacks include *causative attacks*, which influence the learning by altering the training data, and *exploratory attacks*, which alter the classification (i.e. report wrong classification), but do not alter the training [19]. These attacks could be serious for ICS as adversaries may remain in the system causing substantial damage over a long period of time.

1.2 Research Questions

It is critical to national economic resilience that we explore better ways of searching for vulnerabilities and/or weaknesses in the industrial processes and intrusion detection, and understand what the consequences of these vulnerabilities would be if they were to be exploited. Motivated by this gap, this work focuses on answering the following research question:

"Is it feasible to establish a methodology by which vulnerabilities in an industrial process can be established?"

To explore this question, we selected a well-known benchmark, the Tennessee Eastman (TE) Process [20], a modified model of a real chemical process, with a large number of sensors and actuators.

In order to establish the vulnerabilities of the process and the likelihood that evolutionary approaches would be useful in synthesising attacks, the first question that we addressed focused on individual sensors and actuators:

R1: "Can one generate a range of attacks that has a range of physical consequences"

Given an affirmative answer to the first question, the second question then focuses on whether it is possible automatically to identify vulnerabilities in an industrial process control:

R2: "How should one search the attack space effectively and efficiently and identify the most vulnerable components of the process?"

Given the ability to identify attacks, the next logical question to pose is whether it is possible to detect these attacks:

R3: "How should one design a novel intrusion detection system to detect attacks?"

Finally, if it is possible to perform attacks against the defence:

R4: "Can one evolve new attacks against the intrusion detection system?"

1.3 Thesis Structure and Key Contributions

This thesis is composed of nine chapters. Figure 1.3 illustrates the structure of the thesis. Chapter 1 presents our motivation for carrying out this work, and introduces our research questions. In Chapter 2, we review the background material related to attacks against ICS. We introduce the terminology used in the thesis and review the malicious public incidents that have taken place against a wide range of ICS sectors since 1982 to identify: the types of threats ICS are facing; the source of threats; attack vectors; the vulnerabilities of the ICS systems; how systems are initially infected; the intentions of the malicious actors; and countermeasures that are used protect ICS from attacks. Based on examinations of these incidents and the literature, we present an attack taxonomy.

The objective of this thesis is to identify vulnerabilities in existing systems and countermeasures by carrying out attacks against a real-world complex ICS system. The countermeasures we considered are intrusion detection systems. Chapter 3 introduces existing intrusion detection models for ICS and evolutionary computation. There are a wide variety of techniques that are used to detect attacks against ICS, both at the network level and at the process level. However, the details of these studies are often not available or the studies are not mature enough to determine how effective the approaches describe they describe at detecting ICS attacks.

In Chapter 4, we introduce our selected benchmark, a widely used Tennessee Eastman (TE) chemical process control model for along with the threat model for our case study. We consider an attacker that wants to cause some physical damage to the chemical process, but does not have any detailed knowledge about it. We assume the adversary is able to access sensor and actuator signals using vulnerabilities in the ICS, and can manipulate them at some cost to herself. Identifying the most vulnerable components of a large system is cast as an optimisation problem. In Chapter 4, we introduce two evolutionary multiobjective algorithms to employ to optimise attacks and so to use identify the most vulnerable components: those that result in the worst outcomes using the least effort. In this chapter we also selected a number of detection models from machine learning and deep learning that can be

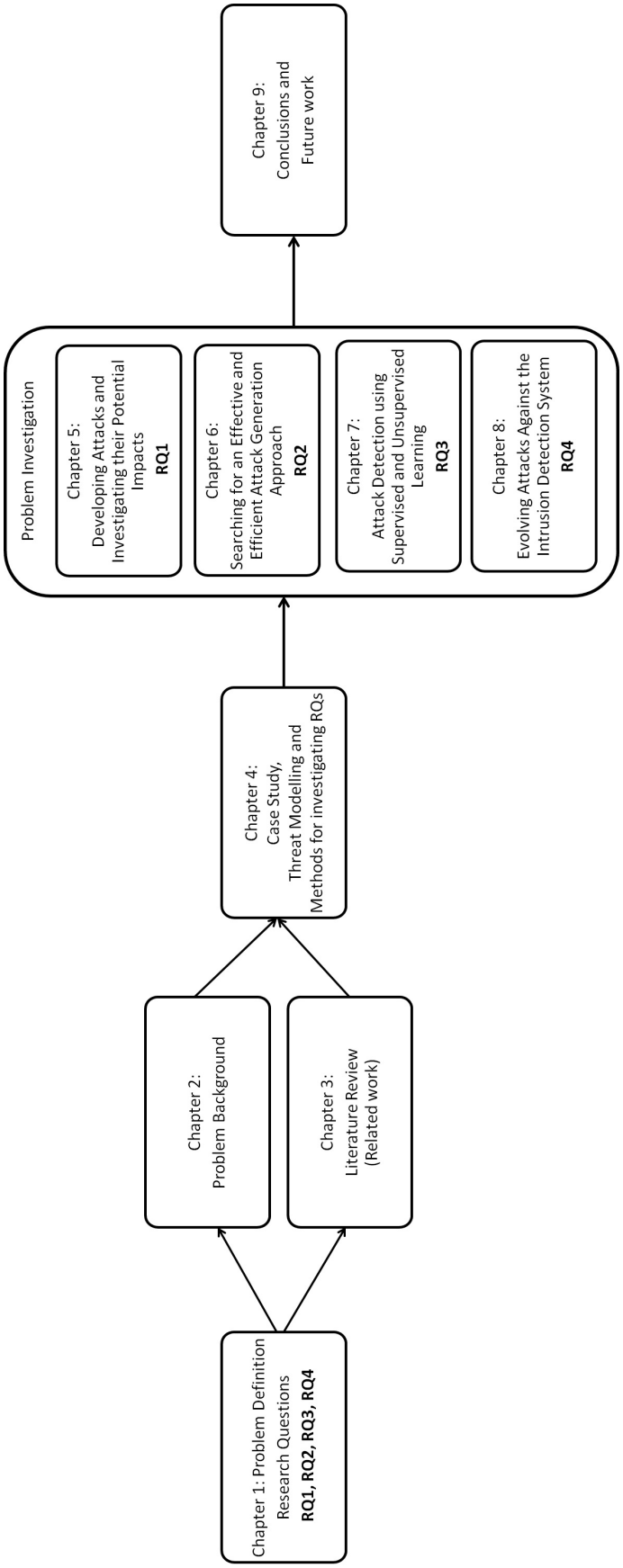


Figure 1.3: Thesis structure

used to detect attacks.

Using the TE process control model and the selected methods, we carried out a set of investigations to address the research questions. We describe these investigations in detail in Chapters 5-8. In Chapter 5, we investigated Research Question 1 by designing a set of attacks and analysing their the impact on the plant. Based on the threat model, we consider DoS, man-in-the-middle and replay attacks, and determine the impact of these attacks on the safety of the system, on the operating cost, and on the production quality. In Chapter 6, we investigate Research Question 2. We use random search, a single objective genetic algorithm, and evolutionary multiobjective optimisation to generate combinations of attacks and report their results. Multiobjective optimisation is based on Pareto dominance, and was implemented using the selection operators of the two reputable evolutionary multiobjective optimisation algorithms. In Chapter 7, we investigate Research Question 3, and implement a range of supervised (Decision Tree, Random Forest, AdaBoost, Support Vector Machines(SVM)) and unsupervised learning methods (One-Class SVM, Autoencoder, Long-Short Term Memory (LSTM) and Gated Recurrent Units (GRU) recurrent neural networks) and determine their performance against the different types of attacks. In Chapter 8, we investigate Research Question 4, and evolve attacks against some of the detection models we implemented in Chapter 7, to identify any remaining vulnerabilities. Finally, in Chapter 9, we summarise our findings, main contributions and discuss possible areas of future work.

The key contributions of this PhD are as follows:

1. Developed an attack taxonomy based on a review of malicious public incidents against in a wide variety of ICS sectors over the past 35 years.

2. Identified a novel area of research for ICS: the development of tools to identify vulnerabilities before they are exploited by malicious actors. It is critical to national economic resilience that we explore better ways of searching for vulnerabilities in industrial processes and evaluating existing countermeasures.

3. Extended an implementation of a well-known chemical benchmark model with security attacks to carry out the set of investigations to address the research questions.

4. Developed an approach based on evolutionary multiobjective optimisation to automate the process of identifying vulnerabilities in ICS, and tested extensively on the complex benchmark model, and against a range of intrusion detection models.

5. Developed and evaluated a range of classic machine learning model and new models from deep learning in their ability to detect attacks.

1.4 List of Publications

Some of the work detailed in this thesis contributed to publications that have been published or, are currently in the submission process:

1. N. Tuptuk and S. Hailes, 'Identifying Vulnerabilities of Industrial Control Systems using Evolutionary Multiobjective Optimisation' (In Submission).

2. N. Tuptuk and S. Hailes, 'Using Unsupervised Learning for Early Detection in Industrial Control Systems', (In Preparation).

3. N. Tuptuk and S. Hailes, 'Crime in the age of the Internet of Things', in Routledge Handbook of Crime Science, Editors: Richard Wortley, Aiden Sidebottom, Gloria Laycock, Nick Tilley, 2018.

4. N. Tuptuk and S. Hailes 'Security of Smart Manufacturing Systems', Journal of Manufacturing Systems, 2018.

5. K. Mrugala, N. Tuptuk, and S. Hailes, 'Evolving Attackers against Wireless Sensor Networks using Genetic Programming' in IET Wireless Sensor Systems, 2017.

6. K. Mrugala, N. Tuptuk, and S. Hailes, 'Evolving Attackers against Wireless Sensor Networks' in Proceedings of the 2016 on Genetic and Evolution-

ary Computation Conference Companion (GECCO '16 Companion), Tobias Friedrich (Ed.), 2016.

7. N. Tuptuk and S. Hailes, 'A survey: Security of Industrial Control Systems', UCL Department of Computer Science, Technical Report, 2015.

8. N. Tuptuk and S. Hailes, 'Covert channel attacks in pervasive Computing', in IEEE International Conference on Pervasive Computing and Communications (PerCom), St. Louis, MO, 2015.

Chapter 2

Background

The number of attacks against industrial control systems, or at least those that have been reported in public domain, is low compared to the number of attacks against public and corporate IT systems. This could be down to some combination of four factors: (i) attacks are hard: they require specialist knowledge of ICS that are not widely present in the community of potential attackers; (ii) attackers shun ICS for their own social reasons: given a similar amount of effort, it may be possible to generate an IT systems attack that has global reach and so generate kudos for the attacker; (iii) people are simply unaware that they are under attack because the effects of those attacks are subtle or because there is a failure to associate the symptoms with the possibility of an attack (as occurred in, for example, Stuxnet); (iv) sharing incidents is not a common practice of the industry: companies tend to avoid making these incidents public to reduce the risk of reputational damage. The evidence from the security community indicates the first two reasons are no longer true. According to a published Symantec security report, the manufacturing sector has become one of those at most risk of attacks [21].

In this chapter we first introduce the terminology used in the thesis. Then, we review the malicious public incidents that have taken place against a wide range of ICS sectors since 1982 to understand the threats ICS are facing. Based on examinations of these incidents and the literature, we present an attack taxonomy covering the source of threats; attack vectors; common exploited vulnerabilities; how systems are initially infected; types of malicious actors and their intentions; and

countermeasures that are used protect ICS from attacks. Some of the text in this chapter has been published in [12].

2.1 Terminology

In this section, we introduce the relevant terminology used throughout this thesis:

- **Availability**:Availability is a security property that ensures the data, system or device is accessible when required. Loss of availability could undermine the industrial process as timeliness of real-time information is often crucial for ICS process.

- **Integrity**: Integrity is a security property that prevents unauthorised modification or introduction of information by unauthorised users or systems. Violation of this property could undermine the safety of the systems.

- **Confidentiality**: Confidentiality is a security property that ensures information is only disclosed to authorised entities. There is a vast amount of sensitive information in ICS including data from sensors and actuators, the plant production process, system specifications, and the chemicals used. Unauthorised access to this information is industrial espionage and has a range of consequences.

- **Safety**: Safety is a critical factor in ICS design to ensure dangerous or hazardous conditions are detected, and appropriate actions are taking on time to bring the system back to a safe condition to prevent consequences that can endanger the environment, people, equipment and the process.

- **Attack**: According to Oxford Dictionaries, the verb to attack is defined as to "take aggressive military action against (a place or enemy forces) with weapons or armed force" [22]. Based on this definition, we define a *cyber attack* as unauthorised action taken against ICS network infrastructure (corporate, control, field), process (software process), system (hardware system, field devices), people, or environment (where the ICS components are lo-

cated or operated) to compromise availability, integrity, confidentiality [9] and safety.

- **Threat**: Following the definition given by the National Institute of Standards and Technology (NIST) [23], a threat is "any circumstance or event with the potential to adversely impact organisational operations (including mission, functions, image, or reputation), organisational assets, or individuals through an information system via unauthorised access, destruction, disclosure, modification of information, and/or denial of service".

- **Threat Actor**: An individual, group or state posing seeking an opportunity to target the industrial control systems to achieve a desired impact.

- **Vulnerability**: Taken from NIST [9], "a weakness in a system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat" actor.

- **Attack Vector**: The means or methods used to collect information, access and carry out attacks on ICS.

- **Attack Impact**: The undesirable consequences or effects of the successful execution of an attack.

- **Countermeasures**: Operational, technical, management or other measures that are designed to reduce the vulnerability of ICS [24].

2.2 History of Attacks

In the following section, a non-exhaustive history of some of the key incidents between 1982 and 2018 is presented to illustrate the vulnerability of ICS, and the attack potential for industrial espionage and physical damage. These attacks illustrate the potential of attacks across critical national infrastructure, plants and factories of all types.

2.2.1 Nuclear Power Plants

Stuxnet [3], first reported in 2010, is believed to be the first worm that was designed with the sole aim of causing physical damage. Stuxnet was different to any other malware seen before in terms of complexity and the advanced set of skills, knowledge and resources needed to implement it [25]. Very little is known about its heritage but it is thought to have been created by or on behalf of a government due to the technical expertise and resources needed perform such an attack. Analysis carried out by Symantec [3] showed that most of the infected machines (approximately 60%) were from Iran. This led the security experts to suspect that the attack was specifically targeting Iran's uranium enrichment facility at the Natanz enrichment plant. Researchers estimated that Stuxnet may have destroyed about 1,000 (10%) of the centrifuges installed at the time of the attack [3].

The malware was designed to attack two models of Siemens PLC (Siemens S7-125 and S7-417) which were controlled by Siemens' Step 7 software. It exploited four zero-day vulnerabilities, propagated itself via removable media that would later be connected to the control systems, and used advanced techniques to mask itself under legal programs to avoid detection. The worm used legitimate certificates, using private keys stolen from two separate companies to sign Windows operating system device drivers [3]. It is not known if Iran was the sole target of Stuxnet, but other countries reported the worm on their ICS equipment including Finland, China and Germany. It is also not known if these wider attacks resulted in any damage, or if they were just used as test cases in preparation for launching the attack against Iran. Stuxnet played an important role in increasing awareness of security for industrial control systems. Prior to Stuxnet, the perception was that control systems, especially those used within nuclear plant facilities, were hard to attack as they were not connected directly to a public network and they operated in isolated rooms with secure physical access. Another perception was that, if an attack was to happen, it would be detected in a timely manner. Stuxnet changed these perceptions, and showed what a highly motivated attacker with the right resources could achieve. Existing countermeasures would have difficulty detecting an attack like Stuxnet.

However, by no means does one need to design a malware as advanced as Stuxnet to cause damage to ICS. Beresford [26] demonstrated inadequate authentication mechanisms on Siemens Simatic S7 PLCs could be exploited remotely to launch a DoS attack and reveal passwords.

There were several other incidents relating to nuclear power plants. In 2008, Unit2 of the Hatch nuclear power plant near Baxley in Georgia was shut down for 48 hours after a software update was installed on one of the computers on the plant's corporate network [27]. The updated computer was used to monitor and control chemical diagnostic data, and the software was used to synchronise data between the computers on the corporate and the control network. After the update, the computer was reset, and the lack of data was interpreted as a significant change in the physical process. This unexpected event caused the emergency safety system to shut down the plant. Although this was not a malicious attack, it illustrates an important vulnerability that system updates are not carried out with security in mind. Furthermore, operators and engineers are not aware of the risks associated with their actions, and they often assume the connection between the corporate network and the control network is secure. In another incident, in 2006 two recirculation pumps that control the flow of the water that goes into the reactor at the Browns Ferry Nuclear Plant in Alabama failed and triggered an automatic shut down of the plant [28]. The pumps failed as a result of the excessive data traffic on the control network generated by a malfunctioning PLC. The details of the incident are unknown; however, it is plausible this might have been due to remote attack.

2.2.2 Petrochemical Plants

An attack against a petrochemical plant in Saudi Arabia was detected in 2017. According to the security researcher investigating the attack, the detected malware was developed to take over the plant's safety instrumented systems controller and cause physical damage; however, due to an error in the code, the malware caused the plant to automatically shut down twice and prompted the plant owner to call for an investigation instead [29]. The malware, now known as Triton (also called Trisis), which was named after the Triconex Safety Instrumented System (SIS) controller it was

targetting, is the first publicly known ICS malware targetting safety instrumented systems [30]. It is thought attackers infected and remained within the petrochemical company's corporate IT network a few years earlier, in 2014, and found a way into the plant's network to accessed the SIS engineering workstation, either by stolen credentials or exploiting an unpatched system vulnerability. Once the attackers accessed the SIS workstation, they were able to shut down the SIS or reprogram it to move the process to an unsafe state and cause physical damage. The examination of the malware showed that the intention of the attacker was beyond causing a shutdown, instead attempting to cause the maximum physical damage. In the worst case, the physical damage could have been involved the release of toxic gas or an explosion putting the lives of people within the plant and in the surrounding area at risk [29]. The actor behind the malware is not known but, given the complexity of the malware, the resources required to test it, and the intended damage, it is likely to be a nation state. According to the Schneider Electric, the distributor of the Triconex SIS controllers, these controllers are widely used in nuclear plants, oil and gas refineries, and chemical plants [31]. Security investigators claim the malware is being used to target companies in North America and others in the world [29].

2.2.3 Wind Turbines and Solar Systems

Problems with the security of wind turbines and solar systems were exposed by an independent German security researcher in the first half of 2015 [32]. The researcher demonstrated the insecurity of several globally deployed energy products: the Nova-Wind Turbines HMI, 442SR Wind Turbine and Sinapsi eSolar Light firmware, and ICS-Cert subsequently issued public warnings [33] and [34]. The authentication credentials used by the HMI were stored in a plaintext file and, therefore, anyone who could access the file could use it to control the HMI remotely and modify the settings of wind turbines. The 442SR Wind Turbine used a web-based system that was vulnerable to cross-site request forgery, and a successful exploit released the authentication credentials (usernames and passwords). The Sinapsi eSolar Light is a system for monitoring and managing remote or local small (less than 50 kWp) photovoltaic plants. There were other authentication vulnerabilities

found in products that are used in control automation systems including monitoring systems [35], ethernet switches [36], clients for remote communications [37], and a PLC programming environment (which lacked authentication mechanisms) [38]. The level of skill required to exploit these vulnerabilities would have been low, and attacks could have been carried out remotely.

2.2.4 Factories

According to an annual IT security report [14] published by the German Federal Office for Information Security (Bundesamt fur Sicherheit in derInformationstechnik, abbreviated as BSI) in December 2014, attackers gained access to a steel factory in Germany and control of a blast furnace. The attackers leveraged spearphishing and other social engineering techniques to gain access to the control network through the corporate network. As a result, the attackers managed to cause unspecified but "massive" physical damage to the system by manipulating individual control components, and bringing the blast furnace under their control. The skill set needed to carry out this attack required knowledge of information security, industrial control systems and production processes [14].

Another key incident in 2014 was Havex/Dragonfly, in which a Remote Access Trojan (RAT) was used to compromise industrial control systems including SCADA, PLC and DCS used within the energy sector across the globe [15]. The aim of the RAT was industrial espionage. Security companies observed targeted spearphishing attempts with PDF attachments against mainly US and UK companies from the energy sector from early 2013 [39]. A Watering-hole attack was used to install the RAT on the machines operating industrial control systems. Multiple legitimate ICS energy vendor websites were compromised and seeded with malware to be included with the intended software download.

In 2011, a sophisticated instance of a RAT known as Duqu [7] [40] infected control systems in Europe (France, Netherlands, Switzerland and Ukraine), Asia (India, Iran and Vietnam) and North Africa (Sudan). The source code showed similarities to Stuxnet from 2010, indicating that the creators had access to the source code of Stuxnet [40]. Unlike the Stuxnet malware, Duqu was not intended to cause

damage. Primarily a RAT, Duqu's payload modules contained remote access capabilities that were used to connect to the command and control (C&C) server and download additional executables, including those used to perform network enumeration, record keystrokes and collect system information. The intention of the RAT was to gather intelligence that could be used to carry out future attacks on industrial control system facilities and other industries. The intelligence collected was encrypted and packed into an empty JPG image file received from the C&C server [7]. Duqu had a number of variants, and made use of C&C servers located in various places including India, Belgium and Vietnam. By default, Duqu was configured to run for 30 days, and then remove itself from the system automatically [7]. However, by adopting a peer-to-peer C&C model, it had the capability to receive additional commands to extend the length of the attack.

In 2005, a worm called Zotob disabled 13 of Daimler Chrysler's auto manufacturing plants across the US, causing them to be offline for 5-50 minutes (a substantial amount of production time), stopping the activities of 50,000 assembly line workers [41]. The worm exploited a buffer overflow vulnerability on a TCP port, found in Windows 2000 systems and some earlier versions of Microsoft Windows, to open a backdoor [42]. According to reports [42], while executing the worm, the operating systems became unstable, resulting in an unplanned cycle of shutdown and rebooting. It is believed that the worm and the new variants of it affected more than 100 companies, including the construction and mining equipment company Caterpillar, news companies CNN and the New York Times, and the airplane company Boeing [43, 44].

2.2.5 Tunnels

In 2013, a Trojan horse targeted the road camera systems of the Israel's Carmel Tunnels in Hafia, causing two outages [45]. The first outage lasted 20 minutes, and the second outage started at the morning rush hour and lasted 8 hours, causing major traffic delays. It is not known who was behind the attack and the details of the malware. Besides causing a significant amount of monetary damage, these tunnels are designed to be used for public shelter in case of emergency. Another attack against

Israel took place on the 6th May 2013 when a hacker group, the Syrian Electronic Army (SEA), sent an email with screenshots of the attack to cryptome.org reporting they had carried out attacks against critical national infrastructure systems in Haifa [46]. The analysis of the screenshots showed the attacks were authentic; however, they had actually gained access to the irrigation control system of the Kibbutz Sa'ar in western Galilee in Israel.

2.2.6 Electricity Industry

In 2007, an ex-employee of the California Independent System Operator (Cal-ISO), the organisation responsible for the California's electricity transmission lines and the electricity market, carried out malicious acts against the organisation [47]. The ex-employee tried a remote attack against the data centre and, when this failed, he entered the facility at night using his unrevoked entrance card, broke the glass cover and pushed the button of the emergency power cut-off, causing disruption and crashing computers.

In 2001, an attack carried out against California's primary electric power grid operator Cal-ISO went undetected for 17 days [48]. Whilst the attack was in progress, there were widespread blackouts affecting hundred of thousands of customers, but Cal-ISO officials denied connections between the blackouts and the attack. According to the investigations, the attack might have been used to gather intelligence about the system and identify vulnerabilities with which to carry out future attacks. It is predicted that attackers gained access to the system by exploiting an unpatched Solaris web server vulnerability. The servers were connected directly to the Internet and they were outside the network's firewall.

In December 2015, one of the first significant, publicly reported, cyberattacks on civil infrastructure was the attack against Ukraine's electric grid [13] resulting in power outages. The electricity distribution management systems of three regional electricity distribution companies that were attacked. Experts examining the attacks found that BlackEnergy malware appeared to have been used to gain entry to the corporate networks of the companies [49]. The method of entry was spearphishing, with malware hidden in Microsoft Word documents with the aim of harvesting

credentials and information related to the ICS systems. There were several variants of BlackEnergy malware [50], which has been used in the past for launching distributed denial of service attacks and espionage attacks ICS networks. A Russian hacking group known as the Sandworm Team (suspected to have links to the Russian Intelligence Service), is known to conduct espionage against ICS networks without causing disruption. The skill set needed to carry out these attacks required knowledge of both IT and ICS operational infrastructure for the electricity distribution such as uninterruptible power supply, SCADA and HMI systems [49].

An attack was carried out against the electric transmission substation in Kiev in December 2016, cutting off electricity for several hours. Unlike previous malware (e.g. BlackEnergy2 and Havex) that were designed for reconnaissance and espionage, the attackers used a malware, *CrashOverride* (also known as Industroyer), that specifically targets ICS to cause physical damage [51]. This was the second publicly known malware (the first was Stuxnet), that targetted ICS to cause physical damage. It had had functionality that was not used during this attack, but could potentially cause longer outages [51]; indeed it is possible that the attackers were testing the malware as a proof of concept [51]. A modular framework, CrashOverride has multiple modules [51]: a main backdoor to access the infected system, install the module to manipulate the system, and to receive new control modules from a remote server; a launcher module to manipulate the ICS and cause physical damage; a spare backdoor in the event the first backdoor gets discovered; and a wiper module to clean up the evidence of the attack. It is thought that the actor behind CrashOverride is the Russian hacking group, Sandworm. The malware is not designed for equipment from a particular vendor in a particular setting, instead it adapts to the electricity grid operations and network communications by connecting to external command and control server to receive new modules. In this way, it can be easily adapted to target other electrical grids around the world.

2.2.7 Oil and Gas Industry

Probably the first publicly known cyberattack against a physical infrastructure is the attack against the Siberian gas pipeline [52]. In 1982, during the Cold War, the

CIA installed a Trojan horse to control systems of the Urengoy–Pomary–Uzhhorod pipeline (also known as the West-Siberian Pipeline), which transports natural gas to Central and Western European markets. The malware covertly changed the speeds of the pumps and valve settings to generate pressure that went above the maximum pressure with which the pipeline joints and welds were able to cope, resulting in a dramatic explosion.

In 2012, the Iranian government confirmed that the computers of their Oil Ministry were infected with malware [4]. The damage this caused is unknown but Iran disconnected their oil terminals from the Internet. The file referencing used by the malware was similar to that used by the Flame malware and, therefore, it is thought that a version of Flame might have been used to carry out this attack. If so, it had the potential to erase data from the hard disks of the computers it infected. Flame, also known as Wiper, has similar complexity to Stuxnet and Duqu with advanced spying capabilities including data gathering, evading detection and maintenance and upgrade [53]. The industry sector targeted by Flame is unclear, but the targets were mainly located in the Middle East and Eastern Europe.

In 2012, a malware known as Shamoon targeted energy companies. Unlike other malware of this kind, Shamoon was not designed for data exfiltration purposes; instead, it was designed to make the infected machines unusable by deleting data and overwriting existing drivers [54]. Saudi Aramco, one of the largest oil companies in Saudi Arabia and RasGas, one of the largest producers of liquid petroleum gas based in Qatar, were among the targets of Shamoon. The damage caused is unknown; however, given the nature of the worm, it is highly likely that data related to the production might have been destroyed. The control systems of other energy companies were also targeted in 2012. The energy company Telvent's firewall and other security mechanisms were bypassed. Telvent reported the attack to customers, and said the attackers installed malware and stole OASyS SCADA files[55]. OASyS is a platform that integrates all the tools used within the control and business level under one centralised system [56]. The intention of the attackers might have been to obtain these files to search for other vulnerabilities with which

to carry out further attacks.

In 2000, the Russian government confirmed that hackers had succeeded gaining access to the systems controlling one of the largest natural gas pipeline networks in the world, the Gazprom pipeline network. According to the Interior Ministry spokesman "some type of Trojan" was used, but the details of the malicious software, and the damage it caused, are unknown [57].

2.2.8 Water Treatment Systems and Canals

On the 8th November 2011, the SCADA systems of an Illinois water utility started to misbehave, causing a water pump to burn out [58]. Initially, the system officers investigating the SCADA log files reported a hack that was traced back to Russia; later, contradictory press releases were made by the officials and government agencies. The FBI and DHS claimed there was no evidence attributing the hack to a Russian IP address [58]. A hacker known as *prOf* claimed he hacked into the HMIs used by the water systems of South Houston (Texas) to illustrate how vulnerable they were. To support his claims, he posted screenshots of the breach. In an interview, the hacker said "this was barely a hack" as the attack consisted of determining the three character long password used for logging into the HMI [59].

In 2000, an ex-employee of Hunter Watertech, the water treatment control system supplier to Maroochy Shire Council of South East Queensland, took control of the waste water management system of the Council [60]. The sewage control system had over 140 pumping stations connected by radio communication to a control centre. The disgruntled employee exploited a vulnerability of the radio channel by masquerading as a controller and sending commands to the pumping station. He was able to stop pumps from running and prevent them from sending alarms to the legitimate controller, causing a release of some 800,000 litres of sewage into the environment, including onto hotel grounds, local parks and rivers, endangering public health, fish and wildlife.

In 2007, an ex-employee of a canal operator, the Tehama-Colusa Canal Authority (TCCA) in California, installed malicious software on the SCADA system that is used to divert water from the Sacramento River and to around California for

agricultural irrigation [61]. The attacker, who had been an employee of the company for 17 years, was fired the day before the incident. The intrusion damaged the SCADA system, which had to be operated manually, causing TCCA financial losses.

2.2.9 Defence Industry

The security wake-up call for Japan was the attack carried out against Japan's major defence contractor Mitsubishi Heavy Industries (MHI) Ltd in 2011 [62]. The attackers used spearpshing attacks to infect 45 servers and 38 computers. Using a collection of viruses, the attacks targeted data on submarines, rockets, missiles and nuclear power stations [62]. According to the reported investigations, carried out by the MHI and the Japanese Police, there was no evidence to indicate that adversaries got hold of any highly sensitive information, but network information such as the IP addresses of systems and devices were compromised [62]. In August 2015, security lab Kaspersky discovered a cyberespionage campaign known as Blue Termite that was specifically targeting Japanese organisations, including heavy industry, chemical, electrical and manufacturing companies [63]. According to the research lab, the attacks had been active at least for the previous two years. It is still active. The attackers leveraged a zero-day vulnerability and a backdoor, tailored to each victim, to obtain sensitive information. Organisations were infected using a number of techniques including, watering-hole attacks and spearphishing attacks.

2.2.10 Traffic Lights

In 2009, an Italian court shutdown the smart traffic lights system known as the T-RedSpeed that had been implemented across Italy and started an investigation that involved over 100 people, including police officers and government officials [64]. They were accused of rigging the traffic lights to stay on the amber light for shorter than the regulated time of 5-6 seconds, and issuing over a million fines to innocent motorists. The engineer in charge of the project was held under house arrest and others were further investigated [64]. Other traffic signs are also vulnerable to security attacks, and have been attacked. In 2009, several states in US were com-

promised by hackers to change the legitimate traffic warning notices to nuisance ones such as "Zombies Ahead" [65]. The details of the attack have not been given, but it is often the case that access panel of the display electronics is simply locked with a padlock and the default password is not changed.

When the traffic engineers in Los Angeles went on strike in 2006, the city decided temporarily to block all engineers from accessing the computers that control most of the traffic signals of the city, to prevent any attacks [66]. However, this measure was not enough. Two of the engineers on strike pleaded guilty to programming the traffic signal system to cause the traffic lights at key intersections to stay much longer on a red light, causing congestion for several days [66] [67]. The engineers programmed the system remotely from a laptop using the code they stole beforehand, and modified the code to prevent others from programming the system back to normal [66]. Another incident from 2006 involving an employee's compromised laptop, was the infection of water treatment system at Harrisbury, Pennsylvania [68]. According to police investigations, the attackers carried out the attack from outside the US. The attack utilised the affected machines as bots to spread spam and other malware. The consequences of this kind of attack could have been catastrophic as the attackers could have used the system to raise the level of chemicals such as chlorine used in water treatment operations.

2.2.11 Transportation Systems

In 2008, the security of transportation systems was questioned when a 14-year-old Polish teenager brought the tram system down in the city of Łódź by changing the track lines [69]. The teenager gathered the necessary intelligence about the system from tram depots, and adapted a TV remote control to switch tram tracks. 12 people were injured as a result of derailing four trams, and other trams had to make emergency stops, causing chaos.

In 2003, the computers belonging to CSX, one of the major rail network operators in the United States, were infected with a virus [70]. The virus shut down the operation of crucial services of the transportation system, including those responsible for dispatching and signalling. The signalling outage affected the entire trans-

portation system, causing major delays and cancellations. Another major incident of 2003 was the Microsoft SQL Slammer worm [71], infecting the Davis-Besse Nuclear Power Plant in Ohio causing network traffic overload on the process control network [72]. It entered the control network through the corporate network, bypassing the firewall via a connection created by a software contractor's firm. When the attack took place, most parts of the plant were offline, and the consequences were not catastrophic, but the Safety Parameter Display System was inaccessible for about five hours and it took an additional six hours to restore the plant process computer.

2.2.12 Ports

In 2001, the web servers belonging to the major US port, the Port of Houston in Texas, were used as an intermediary system to carry out a denial of service attack on another target [48]. The attack exploited a vulnerability in Microsoft's Internet Information Services (IIS) Web Server to send thousands of ping requests, making the server inaccessible to legitimate users [48]. Pilots and shipping companies were unable to access vital navigation and weather information.

In 2013, a group of drug traffickers who hacked into the container management system at the Belgian port of Antwerp to control the movement of containers containing tonnes of cocaine and heroin among legitimate cargoes such as fruit shipped from South America [73]. According to the investigation carried out by Interpol [74], the criminals hacked into the system using a number of methods, over a period of over 2 years. The first breach consisted of carrying out spearphishing attacks to trick employees of the companies into installing malware. Using the malware, they managed to control two of the computers running the port administration system. This breach allowed attackers to change the location and the delivery time of the containers, as well as the security details required to pick up the containers. In this way, drug traffickers were able to send their drivers to steal the shipments before the legitimate drivers came to pick them up. When these breaches were found, criminals carried out other attacks by breaking into the premises housing these systems and installing hardware such as keyloggers and tiny computer boards to access

secure data and manage attacks remotely[73].

2.3 ICS Attack Taxonomy

On the basis of the examination of previous attacks, and the common techniques used against IT networks, the taxonomy shown in Table 2.1 was devised. In the following subsection, we explain the components of the taxonomy.

2.3.1 Threat Origin

On the basis of the examination of previous attacks, the sources of those attacks would appear to come from a wide variety of actors:

- **Foreign Intelligence Services/Nation States:** Nation-state sponsored attacks are sophisticated attacks and their potential to cause damage is high. The originators have significant power to finance, obtain necessary resources and mobilise people with advanced skills, and influence vendors to modify software or hardware systems/devices and install malware or backdoors with which they can carry out these attacks.

- **Rivals:** Rival organisations or companies may carry out attacks to damage reputation or for industrial espionage to steal intellectual property.

- **Terrorist Groups:** Terrorist groups may want to attack the operation of ICS to threat national security, cause causalities, damage the economy and create fear.

- **Organised Crime:** Attacks by an organised criminal network are usually motivated by financial gain. Criminals are currently very active in the online world. So, for example, attacks on control systems may be carried out to extort money by threatening asset owners for ransom [75].

- **Thieves/Vandals:** Electricity substations have been an attractive target to metal thieves who steal steal copper cables from stations and other structures [76] [77], costing the UK economy an estimated £770 million a year [78]. While trying to take metal out of substations, thieves risk their lives: in 2010,

six people in the UK died while stealing copper cables [76]. Given their determination and the substantial revenue they can generate from metal theft, it is plausible to suggest that vandals might try to hack ICS to reduce the risk of harm.

- **Hacker Hobbyist:** Attacks carried out by hacker hobbyists are typically motivated by curiosity, the thrill of doing something not permitted, or the desire for recognition and status. These attacks are generally unsophisticated, often exploiting vulnerabilities that are public, but that can still cause substantial damage.

- **Hacktivists:** Hacktivists are hackers that are often driven by a political idea; hactivism is used as a form of online activism [79]. The motivations for their actions may be anything from defence of free speech to an anti-nuclear stance. Over the past several years, two powerful hacking collectives, Anonymous and Lulzsec, have carried out a large number of attacks on the Internet. These attacks include support for protests in Iran [80], protesting against the Australian Government for Internet filtering legislation and web censorship regulations [81], compromising web sites and email of oil and gas companies to protest against oil prices, and bringing attention to WikiLeaks and other political causes [82]. Security services with responsibility for critical national infrastructure have also been targeted by hacking communities [83].

- **Insiders:** An insider could be a company executive, current or former employee, business partner, contractor, service provider, vendor, guest, support staff (e.g. cleaners, system support technicians) "or someone else who has a formal or informal business relationship with an organisation" [84]. The most common insider attacks are: unauthorised access to, and use of, corporate information; unintentional exposure of private or sensitive data; virus, worms or other malware; and theft of intellectual property [85]. However, threatening insider behaviour occurs in many contexts and appears in various forms, and, unfortunately, only becomes public if legal action is taken against the

attackers. This is not often the case due to concerns about negative publicity, being unable to identify the individual/s committing the act, and lack of evidence [86]. The motives behind insider attacks are very diverse, [87] including financial gain [88]; problems at work; nationalistic/ideology (serving another nation or political cause); fear or coercion; and excitement/challenge (the thrill of doing something not permitted, or the desire for recognition and status). Not all apparent attacks from insiders are malicious: insiders may cause unintentional exposure of sensitive data or systems by error or misuse. Furthermore, rules are often broken due to deadlines, lack of awareness, unusable or ineffective policies or procedures.

2.3.2 Threats

On the basis of previous attacks there appears to be two primary motivations behind attacks aimed at industrial control systems: 1) exfiltration: harvesting sensitive information 2) sabotage: disrupting control performance:

- **Data Exfiltration:** Adversaries may carry out malicious actions to gather intelligence about a target, either as a preliminary step to allow for the design of further attacks (reconnaissance), or for industrial espionage. This may involve identifying network infrastructure, insecure applications, unpatched systems, process implementation details and other sensitive information (including credentials). These attacks can be carried out against the corporate network to acquire knowledge (e.g. operating system and versions, unpatched system vulnerabilities, firewall rules, or whether the control network is reachable). The primary objective is usually to obtain the credentials needed to infect other systems and devices, and so to *propagate and escalate privileges* to gain access to the actual control network. Data exfiltration can also be carried out against the systems and devices at the control level to harvest sensitive information related to the system (known as the operational technology) including the software used for programming control devices, operation of the safety system, network communication interface and protocols and any other

Threat Actor	Threat Type	Attack Vector	Vulnerability	Initial Infection	Impact	Countermeasures
Foreign Intelligence Services/Nation States Rivals Terrorist Groups Organised Crime Thieves/Vandals Hacker Hobbyist Hacktivists Insiders	**Data Exfiltration** **Direct Control:** Reprogramming control devices Modifying control setting/configurations **Indirect Control:** Modifying control data in transmission or storage Denial or delaying flow of control data	Eavesdropping (Scanning, Probing Sniffing) Denial-of-Service Impersonation Man-in-the middle Replay data Delay data Drop data False data injection Data tampering Side-channels Covert-channels Physical access Blended attacks	Zero-day vulnerabilities Insecure architecture Insecure communication protocols Backdoors and holes in network perimeter Inadequate configuration and maintenance Poor policies and procedures Poor physical access	Social engineering Drive-by-download Hacking the supply chain Insiders Physical access	**Direct Impact:** Process disruption Resource-exhaustion Data disclosure Data destruction Damage to production Damage to equipment Damage to safety Damage to environment Injury/Loss of lives **Indirect Impact:** Loss of reputation Economic damage Public response Legal prosecution	**Preventive**: Network Perimeter Control Authentication/Access Control Cryptographic protection Software Updates Vulnerability scanning Usable Security Policies Awareness/Training Physical Security Supply Chain Protection **Reactive:** Malware/Antivirus Software Network Monitoring Process Anomaly Detection **Response:** Network reconfiguration Device/System Isolation Industrial Safety Systems

Table 2.1: Attack taxonomy for Industrial Control Systems

knowledge related to the control and countermeasures.

- **Direct Control:** These are attacks against the process by either *reprogramming control systems or field devices* (e.g. modifying setpoints, process values, sending false system status to operator machines) or *modifying control settings or configurations* (e.g. modifying safety thresholds/alarm, introducing rogue devices) [9].

- **Indirect Control:** These are attacks against the integrity of the control data by *modifying the data in transmission*(e.g. modifying the data sent by sensor or controller); *tampering with data in storage*; or *denial or delaying flow of control data* (e.g. stopping or delaying new data reaching the controller or the actuators).

2.3.3 Attack Vectors

A wide variety of techniques have been used to carry out attacks on ICS:

- **Eavesdropping:** This attack involves obtaining sensitive information related to the ICS such as: identifying devices, services or vulnerabilities on the network; or extracting password and other information from network traffic to perpetrate further attacks. Eavesdropping techniques include :*scanning* the network to identify possible methods of entry (e.g. open ports); *probing* the network by sending packets to target devices and monitoring their response; and *sniffing* the network to capture, analyse and monitor network traffic to gain sensitive information.

- **Denial of Service (DoS):** To deny the availability of a system to legitimate entities by *flooding* the targeted system with traffic.

- **Impersonation:** This attack involves impersonating a device, system or an entity to gain unauthorised access to a network or a host. Because many ICS communication protocols do not provide mechanisms for authenticating the source or destination of a message, ICS systems are vulnerable to: impersonation attacks such as IP and ARP spoofing; and communication hijacking.

Inadequate authentication control mechanisms mean that entities can masquerade as one another by falsifying their identity to gain illegitimate access, and so to steal or modify data, spread malware or carry out DoS and man-in-the-middle attacks.

- **Man-in-the-middle (MitM):** In this attack, the attacker sits between communicating devices and alters the communication between them.

- **Replay data:** Replay is a specific, but common, form of MitM attack in which old, but valid, network data are retransmitted by the attacker.

- **Delay data:** The attacker delays the network data between the sender and receiver (e.g. causing a controller to receive old data from a sensor). Delays may cause operational errors, inefficiencies or cause the system to fail.

- **Dropping data:** This attack happens when a compromised node selectively drops or deletes some of the network data (such as data packets) in wireless industrial control networks.

- **False data injection:** This attack is a deception attack, injecting bogus or malicious data into the network. This is different to MitM as it does not alter the traffic between communicating devices.

- **Data tampering:** Data tampering involves unauthorised modification of data in storage or at rest, for example through device configuration, settings or programming code.

- **Side-channels:** Attackers use a variety of techniques to gain information about a system or a device by analysing apparently incidental sources of data such as power consumption, light emissions, traffic flow, timings, electromagnetic, acoustic and thermal emissions from hardware components [89].

- **Covert-channels:** This is an attack that relies on device compromise. Given this, an attacker uses a channel that was not designed for data transfer to leak

confidential information from a secure environment (bypassing the existing security measures) [90].

- **Physical access:** Attackers with physical access to a system can modify the location of devices (*node displacement*); can decalibrate devices such as sensors to modify input signals, or can exploit physical properties of the devices (e.g. through glitch attacks involving modification of the clock or the power supply to the chip to manipulate the operation of the system). Moreover, they can simply install physical nodes (*rogue device*) to carry out passive attacks (monitor the physical process) or more active attacks (act as an controller); and can install malware via infected removable media such as USB memory sticks.

- **Blended attacks:** Blended attacks consist of the use of multiple attack vectors packed as exploit kit or another form of malware that will be installed on the target infrastructure to exfiltrate data and/or sabotage the system.

2.3.4 Vulnerabilities

On the basis of the examination of previous attacks and NIST's Guide to Industrial Control Systems Security [9], the categories of vulnerabilities are as follows:

- **Zero-day Vulnerabilities:** These are previously unknown vulnerabilities in software or hardware. Once these vulnerabilities become public, it can take a while for the software developer or hardware manufacturer to provide a patch, and longer for it to be installed. Once these vulnerabilities are disclosed, the number of attacks exploiting them may actually increase since more attackers are able to launch the attack and do so in an attempt to find unpatched systems [91].

- **Insecure architecture:** Traditionally, industrial control systems have been been designed without security in mind, or with the explicit presumption that the system is isolated and so not subject to attack. Bringing security into existing systems or evolving systems (e.g. adoption of COTS products, Industry

4.0) could create vulnerabilities if security controls are not not deployed and configured for new changes in the architecture.

- **Insecure communication protocols:** Industrial Control Systems make use of a wide variety of communication protocols that may have no or limited security mechanisms for authentication, confidentiality and integrity. Adversaries can exploit these vulnerabilities to monitor unecrypted communication between devices, replay, modify or spoof data or introduce rogue devices into the network [9].

- **Backdoors and holes in the network perimeter [9]:** A backdoor is any mechanism that bypasses normal security controls in order to access a computer system. Known as hidden-accounts, backdoors have often been deliberately inserted into the ICS infrastructure for purposes such as ease of maintenance and troubleshooting by vendors. These backdoors can be exploited by adversaries to access the ICS. Similarly, there could be vulnerabilities related to network design, such as weak network segmentation, non-existent or inadequate firewalls.

- **Inadequate configuration and maintenance:** The components of ICS infrastructure, including hardware, firmware and software need to be configured and maintained throughout their lifecycle. Inadequate configuration and maintenance, such as using default settings (e.g. default passwords, connections or settings not changed); using systems and applications that are no long supported (updates not available); not patching on time; deploying new security measures without testing; leaving unused ports and protocols open; poorly maintained access control mechanisms; and lack of, or outdated, intrusion detection systems can be exploited by the adversaries.

- **Poor policies and procedures:** Inadequate security policies and procedures open the door to vulnerabilities. All countermeasures should be traceable to a policy to ensure accountability and uniformity [9]. Security policies and procedures should be documented, updated, enforced and usable. If secu-

rity policies and procedures become a burden to staff, history suggests that they will be unwilling to comply with these policies and procedures and may misuse the system deliberately. Lack of formal ICS security training and awareness will also create an insecure ICS environment, and may make staff vulnerable to social engineering attacks.

- **Poor physical access:** Weak physical access can lead to physical tampering with ICS systems and devices; installing malicious malware (including keyloggers), and making unauthorised changes to the system and operating environment of the ICS.

2.3.5 Initial Infection

Advanced attacks against industrial control systems, such as Stuxnet, Havex, BlackEnergy and Triton, required attackers initially to install malware on the target infrastructure. This step is known as the initial infection. Initially the purpose of the malware could just be data exfiltration at the corporate level but this might be followed by scanning other systems and propagation. Methods used for initial infection include the exploitation of *insiders*; the use of *social engineering techniques* (e.g. malicious attachments, spearphishing); *drive-by download* (the use of vulnerabilities in web servers, web browsers and browser plugins to install malware on users' computers without their consent); *hacking the supply chain* (e.g. watering hole attacks in which malware is placed with the original software updates from a compromised trusted vendor website); and *physical access*.

2.3.6 Attack Impact

Impact refers to the consequences of carrying out successful attack against ICS. These can be divided into two main categories: direct impact and indirect impact. *Direct impact* is tangible and immediate, including process disruption; *damage to production, equipment, safety and environment*; *data disclosure*; *data destruction*; and *injuries and loss of life* [9]. *Indirect impact* is often hard to define and quantify since it frequently aims to have long-term consequences such as *damage to reputation and business relationships*; *economic damage*; *legal prosecution*; and *loss of*

public confidence [9].

2.3.7 Countermeasures

Table 2.1 lists some of the common types of countermeasures that are used to protect ICS from attacks. Existing countermeasures [9] can be divided into three main categories: *preventive*, *reactive* and *response*. *Preventive* defence measures include restricting access to the ICS network through: *perimeter control* such as firewalls and demilitarised zones; *authentication and access control mechanisms* for users of the ICS; *cryptographic protection* to achieve confidentiality and integrity of data; *active scanning* of systems, devices and the network for vulnerabilities; *software updating*; the development of *security policies* that are feasible and usable and that define roles and procedures for maintaining security; *personnel awareness and training programme*s to avoid misuse of systems the enforcement of physical security; and measures for *protecting the supply chain*. *Reactive* defence measures include: *intrusion detection* at the host and network level, including *monitoring for unusual behaviour and malware*; and *anomaly detection at the process level*. After an attack is detected, *response* measures may include: *reconfiguring the network*, including removing or redirecting network traffic and blocking ports; *isolating or removing certain systems or devices* from the network; and the development *industrial safety systems* that incorporate security measures to protect the safety of the process, equipment and personnel, such as the automatic shutdown of equipment and the raising of alarms when hazards are detected.

2.4 Summary

Although some of the more sophisticated attacks carried out against ICS required a vast amount of knowledge and resources, the attacks discussed here show that it is not sensible to rely on this assumption remaining true for the indefinite future: adversaries are using a variety of techniques to sabotage ICS and to exfiltrate data. As past attacks show, the potential consequences of security attacks on ICS include injury, death, and damage to physical infrastructure, equipment and environment. It is becoming increasingly vital that we identify the vulnerabilities in ICS before they

are exploited by threat actors. In the next chapter, we review the related work for detecting attacks, and possible methods that can be used to identify vulnerabilities before they are exploited by malicious actors.

Chapter 3

Literature Review

In this chapter, we discuss the existing research related to our study, including current approaches to the detection of new vulnerabilities. We first introduce existing work related to intrusion detection systems, and then introduce a potential tool that can be used to search for new vulnerabilities, evolutionary computation. Previous work related to generating attacks while evading detection is examined.

3.1 Detecting Attacks on Industrial Control Systems

Security is a dynamic process that requires proactive measures to manage attacks that are continuously evolving. There will always be some vulnerabilities, and attackers will continue to search for vulnerabilities to exploit. Response is needed to detect and discourage these attempts. In this section, some of the existing academic approaches to intrusion detection in industrial control systems are discussed.

3.1.1 Performance Metrics for Intrusion Detection Systems

In traditional IT systems, the performance of intrusion detection systems (IDS) are measured and reported using the standard metrics including accuracy, precision, recall (true positive rate), F1 score, false positive rate and false negative rate, defined as follows [92]:

$$Accuracy(A) = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.1}$$

$$Precision(P) = \frac{TP}{TP + FP} \tag{3.2}$$

		Actual	
		Attack	Not-Attack
Predicted	Attack	True Positive (TP)	False Positive (FP)
	Not-Attack	False Negative (FN)	True Negative (TN)

Table 3.1: Confusion matrix

$$Recall(R) = \frac{TP}{TP+FN} \tag{3.3}$$

$$F_1 = 2\frac{P \times R}{P+R} \tag{3.4}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP+TN} \tag{3.5}$$

$$\text{False Negative Rate (FNR)} = \frac{FN}{FN+TP} \tag{3.6}$$

Their definitions are as shown in Table 3.1, and they are as follows:

- True Positive (TP) indicates an attack signal correctly labelled as an attack.
- True Negative (TN) indicates a not-attack signal correctly labelled as a not-attack.
- False Positive (FP) indicates a not-attack signal incorrectly labelled as an attack.
- False Negative (FN) indicates an attack signal incorrectly labelled as not-attack.

Accuracy measures the proportion of instances that were assigned to the correct classes. *Precision* measures the proportion of the positive results that are identified as positive (how many of the attack indications are really attacks). *Recall* (also known as the true positive rate (TPR)) measures the proportion of actual attack signals that are correctly identified as such. The *F1* score considers both false positives and false negatives, and it is calculated using the weighted average of precision and recall [92]. Like all the other measurements, the best value for F1 score is 1 and the worst is 0.

3.1.2 Intrusion Detection for ICS Network and Hosts

IDSs are generally classified by detection technique and by audit source. The former is often classified into two groups: *knowledge-based* or *behaviour-based*, and the later *network-based* or *host-based* [93]. Host-based IDS are based on a single node and, therefore, make use of data maintained by that node to identify unauthorised behaviour. Network-Based IDS often have a single node that is dedicated to collecting and analysing the network activity to detect intrusions. Thus, they are based on models of network traffic or host (device or software application) behaviour. The majority of existing intrusion detection studies focused on industrial control systems are also related to detecting intrusions by analysing host/application level access and usage of network communication protocols.

Knowledge-based intrusion detection systems, also known as pattern or signature-based detection systems are based on collecting knowledge about previous well-known attacks, and designing signature rules specifying intrusion-specific patterns. Data from events, such as patterns of network traffic, are compared with these specific signatures to detect the presence of intrusions. Proposed IDS often focus on designing attack signatures for detecting unauthorised access to networks and hosts [94] as well as detecting DoS and spoofing attacks [95]. A major drawback of these systems is that they are only able to detect attacks if their signatures are available. Furthermore, a world in which attackers are constantly changing their attack vectors to avoid these IDS systems means that the identification and update of signatures also becomes a constant activity.

Behaviour-based intrusion detection systems also known as anomaly-based systems, look for deviations from normal system behaviour. An understanding of what is normal is derived from learning the behaviour by employing unsupervised learning or semisupervised learning under conditions that are believed to be attack-free. Most of these approaches make use of data related to traffic between ICS components. For example, in [100], a neural network is presented that is trained on normal network traffic data with the aim of detecting abnormal network traffic between ICS components. A semisupervised anomaly detection system based on measur-

Specific Domain/Reference	Detection Principle	Audit Source	Attack Vector	Data Source
Smart grid [94]	knowledge	host	impersonation, tampering	testbed/unreleased
Smart grid [95]	knowledge	network	sniffing, impersonation, DoS	simulation/unreleased
N/A [96] [97]	behaviour-specification	network	DoS, probing	testbed/unreleased
N/A [98]	behaviour-specification	network	false data injection, delay, replay	testbed/unreleased
N/A [99]	behaviour-specification	network	tampering, replaying	operational/unreleased
N/A [100]	behaviour	network	unknown	simulation/unreleased
N/A [101]	behaviour	network	19 CVE attacks (tampering, injection, masquerading)	operational
Smart grid [102]	behaviour	network	tampering, denial, delaying	no dataset
N/A [103]	behaviour	network	impersonation, DoS, MitM	simulation/unreleased
N/A [104]	behaviour	network	false data injection	operational
Industrial sensor networks [105]	behaviour	network	DoS, eavesdropping, impersonation, MitM	operational
Smart grid [106] [107]	behaviour	host	DoS, sniffing, probing	KDD'99, NSL-KDD (public)
Smart grid [108]	behaviour	network	replay, injection, zero-day	HIL simulation
N/A [109]	behaviour	host	false data injection, DoS, MitM, replay	operational/unreleased
N/A [110]	behaviour	network	scanning, tampering, DoS	testbed/unreleased
N/A [111]	behaviour	network	scanning, tampering, injecting, MitM, DoS	testbed/unreleased
Water treatment [112], [113],[114] [115]	behaviour	process	36 MitM type of attacks	testbed/public
Biochemical process [116]	behaviour	process	MitM, tampering	no dataset
N/A [117],	behaviour-specification	process	eavesdropping, MitM, tampering	simulation/unreleased
Gas-oil plant [118]	behaviour	process	MitM	simulation/unreleased
TE chemical plant [119], [120], [121] [122]	behaviour	process	DoS, MitM	simulation/unreleased
TE chemical plant [123]	behaviour	process	DoS, MitM	simulation/released

Table 3.2: Some proposed IDSs for ICS

ing similarities (using Manhattan and Euclidian distance) between transport-layer packet payloads is introduced in [101]. The work carried out by Zhang et. al [107] [106] proposes a distributed IDS based on support vector machines and an artificial immune system to detect and classify attacks for smart grids, utilising data from systems and devices in multiple network layers of the grid network (home area network, neighbourhood area networks, wide area networks). One of the assumptions that IDS models in Industrial Control Systems often make is that the traffic from these networks is deterministic because there is a static network with well-defined protocols and strict timing requirements. Examples of this behaviour are sensors that may send data to a controller at certain times. Leveraging the determinism of the information cycle, a semi-supervised model is proposed to detect anomalies by auditing sensor and actuator data [102]. Some researchers have proposed models that combine the advantages of signature-based detection and behaviour-based detection to determine whether a system is under attack. In [96], authors propose a model that monitors the Modbus/TCP protocol function codes and lengths with a Bayesian detector, and uses rules to detect deviations from expected communication patterns. In their subsequent work, the authors developed a visualisation tool that can be used to analyse network patterns [97]. In [98], authors present another IDS system designed for detecting anomalies in Modbus/TCP networks using a finite state machine model.

One of the limitations of IDS studies in ICS is lack of realistic testbeds and datasets. In [109], authors use an ICS testbed that contains SCADA systems for a water storage tank control system to test their intrusion detection systems. Motivated by the lack of authentication support in ICS communication protocols (e.g. MODBUS, DNP3 and EtherNet/IP), they make the assumption that anyone with access to these systems can carry out a number of attacks: false data injection attacks (injecting false commands to devices such as RTU, MTU), sending false responses to master devices and denial of service attacks. They designed a supervised classifier using neural networks to detect these crafted attacks.

A list of some of the IDS systems designed to detect attacks or anomalous

behaviour by monitoring the traffic at the network or host level is given in Table 3.2. A fuller list of IDS systems designed for cyber physical systems can be found in [93]. The data source used to evaluate the effectiveness of the various IDS is one of three types: i) operational data from real-world systems; ii) testbeds (real-world systems designed for experimentation); iii) simulation (including co-simulation and hardware-in-loop). In general, the studies tend to show limitations that result from a lack of common testbeds and open datasets: there are oversimplified test cases and threat models; there is a lack of reported performance metrics; and evaluation is sometimes subjective. At the same time, sophisticated attackers are deploying mechanisms to evade detection. This is done by making use of the vulnerabilities of the detection systems to hide their actions, making the detector ineffective. In terms of IDS, this means avoiding actions that would trigger signature detectors or using actions that are hard to distinguish from the normal behaviour of the plant. Detection that adapts to an adaptive adversary is an area of IDS research that has not been studied extensively for ICS applications.

3.1.3 Anomaly Detection in Process Control

Detecting attacks against network communication protocols is part of the solution, but IDS that only do this fail to identify attacks against the process control itself. The focus of our research is precisely these later attacks. To counter these cases, researchers in [116] and [117], proposed IDS systems that can detect attacks on process control. McEvoy and Wolthusen [116] proposes a causal model that is used to define the relationship between sensor readings using non-linear structural equations. They use a brewery bulk and fill pasteuriser as a case study to evaluate their approach but, as pointed out by the authors, this approach requires extensive modelling of the system. Hadziosmanovic et. al [117] propose an approach that extracts model variables from the ICS network traffic, characterises the extracted variables into relevant categories and then creates an autoregressive model to capture the behaviour at the process level. They test their model on network data (not open-source) collected from two real-world water purification plants.

Although detecting attacks against ICS processes has received little attention

in the information security field, fault detection and diagnosis for industrial processes is an active research area that has been widely studied by the control community. Faults are defined as abnormal process deviations from the standard behaviour and are caused by equipment failing, human errors, signal interrupts and any other anomalies. Essentially, these approaches have been proposed both to detect anomalies (fault detection) in process data and to identify where the anomaly occurred (fault diagnosis). As approaches, they are equally applicable to cyber attack detection in industrial control systems.

As a benchmark, the Tennessee Eastman (TE) Chemical process model [20] has been broadly studied for fault detection and diagnosis. The TE process model comes with 21 programmed disturbances identified as faults. The simulation delivers multivariate timeseries data containing 53 variables: 41 process measurements and 12 manipulated variables. Training datasets can be gathered using the Fortran simualtion code provided by Tennessee Eastman company, or by using the MATLAB/Simulink model that was implemented in 2002, and updated by the Chemical Engineering Department, University of Washington. Both original Fortran and MATLAB versions are available at [124]. However, most of the existing studies use the Braatz Dataset [125], which has been made available by MIT's Braatz Group, from the Process Systems Engineering Laboratory, and that was generated using the Fortran code. This dataset contains a training dataset and a testing dataset. These are fairly small datasets: each fault is described by a 480x52 matrix; and testing datasets are of size 960x52 (rows representing different measurements, columns representing different features).

Early studies on the TE process model focused common dimensionality reduction techniques such as Principal Component Analysis (PCA) [126] [127] [128], Dynamic Principal Component Analysis (DPCA) [129] and Canonical Variate Analysis (CVA) [129], with distance metrics such as Hotelling's T-squared distribution (T2) and Q statistics used to detect faults. Others have proposed models to improve the performance of classic PCA using various techniques such as nonlinear dynamic principal component analysis [130], multiway principal com-

ponent analysis (PCA) [131] and PCA-wavelets [132]. Bakdi and Kouadri [133] proposed a model using PCA with a modified exponentially weighted moving average (EWMA). Authors in [134] applied Discriminant Partial Least Squares (DPLS) and argued that it improves the fault diagnosis on small scale classification problems compared to classic Fisher's Discriminant Analysis (FDA). Misra et al. [135] applied a multi-scale PCA to a multivariate process on an industrial gas phase tubular reactor system. Later models focused more on utilising machine learning techniques such as support vector machines (SVM). Chiang et al. [136] showed SVMs were better at classifying faults on the TE process in comparison to Fisher Discriminant Analysis. In [121], the authors integrated PCA with a neural network and tested their approach using both the TE process model and a continuous cast steel slab process. In [137], Kernel Independent Component analysis (KICA PCA) and SVM were used to diagnose on faults on the TE process. In [138], the authors propose a One-Class SVM to detect faults in Heating, Ventilation and Air Conditioning (HVAC) chiller systems. Mahadevan and Shah [139] applied One-Class SVM with recursive feature elimination to the TE process model for fault detection and diagnosis, and report that SVM outperforms PCA and DPCA. In [140], Gao and Hou used multi-class SVM to predict the status of the TE process by first applying PCA to reduce the dimensionality of the data. The authors used grid search, genetic algorithms and particle swarm optimisation to select the parameters for the SVM. Chen et al. [141] proposed cognitive fault diagnosis, in which they assume no prior knowledge of faults. The authors also propose "learning in the model space" [142], in which they map the signal space into a reservoir model space and then apply incremental single learning algorithms to this model to detect faults. They carried out experiments in both supervised settings using supervised machine learning algorithms (e.g. Classification and Regression Trees, SVM, NaiveBayes, ensemble algorithms), and cognitive settings, using unsupervised algorithms (e.g. K-means, HCluster, One-Class SVM).

The dataset used for all these approaches is very small and is not representative of the big data problem the industry is facing. Furthermore, most of these

approaches assume faults are known prior to training. Unlike these, Marti et al. [143] proposes using a time series segmentation method to segment the dataset into homogeneous subsets that can be analysed in a separate fashion, and use one-class support vector machines on unlabelled normal operational data to learn to detect attacks. To test their approach, they applied it to a real-life (private) dataset obtained from over 250 sensors attached to an operational oil turbomachine in Brazil.

Some deep learning approaches have also been used for anomaly detection. In particular, recurrent neural networks have been applied to range of complex sequential problems. In [144], a Long Short Term Memory Network (a type of recurrent neural network) was tested on two proprietary real-life engine datasets and three publicly available datasets (ECG, space shuttle, power demand); however, the focus of this work was univariate time series prediction. Lv et. al [145] used unsupervised deep learning (an autoencoder) to learn features of the data, and then applied a deep learning softmax classifier to classify faults. Zhao et al. [146] proposed a fault diagnosis method based on LSTM, and evaluated it on the TE process model using the Braatz dataset.

The security community also carried out a number of studies in an attempt to detect attacks on the industrial control processes. Table 3.2 lists some of these detection models. We have not added the studies related to the fault detection and diagnosis to this, as most of these studies are based on known faults. Although this is still a poorly researched area, these studies focussed on the TE process model, and the dataset (available upon request) from the Secure Water Treatment (SWaT) testbed at Singapore University of Technology and Design [147]. SWaT is a fully functional version of a water treatment plant designed for cybersecurity research. The dataset is a multivariate time series dataset like the TE process, containing normal continuous operational data (7 days) and data collected under attack scenarios (4 days) containing 36 different network attacks [112]. In [113], the authors compared unsupervised LSTM based deep neural network and One-Class SVM, and showed that LSTM performed slightly better than One-Class SVM. Kravchik and Shabtai [114] provide further research on a variety of DNN architectures (convo-

lutional and LSTMs) of unsupervised DNNs to detect attacks on SWAT test data. In [115], authors also used LSTM to detect attacks on the SWaT testbed, but used Cumulative Sum control Chart (CUSUM) to calculate the deviations, and so to determine anomalies.

If not the first, one of the earliest studies that looked at the security of ICS using the TE process model as a case study was by Cardenas et al. [119]. They introduced an attack model (DoS and integrity attacks), and used a variant of the TE process model obtained by linearising the (non-linear) TE model, coupled with a non-parametric CUSUM to detect anomalies. Filonov et. al [118] created their own model using part of a real gasoil plant. They generated attacks on the model, such as modifying temperature and tank flow levels. They developed an unsupervised LSTM model using the normal multivariate time series data to detect anomalies. The same team applied another variant of RNN, a Gated Recurrent Units (GRU) model on the normal operation of the TE process model [123] to detect attacks. They used their own implementation of the TE process (not open-source), implemented in Python, and simulated cyber attacks (DoS, Integrity, Noise) and generated data for the model. They report that their model performs better than dynamic PCA. Similarly, Keliris et al. [120], extended the TE process model to a Hardware-In-The-Loop testbed, and simulated their own cyber attacks (integrity, DoS). They used a supervised machine learning algorithm, SVM, to detect attacks. The authors do not report performance metrics. Kiss et al. [122] extended and carried out cyber attacks (replay, DoS, integrity) against the TE process, and applied a clustering method based on Gaussian mixture models to detect attacks. They also do not report performance metrics, only indicate whether the attack was detected or not.

In this section, we covered a number of anomaly detection systems that were designed to incorporate the behaviour of the process under control. Machine Learning techniques such as SVM and deep neural networks techniques, in particular the newly developed recurrent neural networks, show promise for further research to secure ICS, and we will focus on investigating these techniques in the design of our anomaly-based intrusion detection. Unfortunately, making a consistent com-

parison with existing research is not possible due to limitations of reported results. In general, not all studies report clear detailed descriptions of performance metrics, including true positive rate and false positive rates. The security community is more at fault in this than fault detection community. Detection latency is also an important metric, and researchers rarely report this. We found only two studies that report it from the fault detection and diagnosis community [129] [139], and a single one from the security community [123]. A common metric is essential to achieve a consistent way to evaluate and compare future studies.

As a benchmark, the TE process model has been widely studied, because there are few other realistic nonlinear process models. Fault detection and diagnosis studies tend to use the open-source fault data generated by the Braatz Group; however, this dataset may not be suitable for security studies. As discussed earlier, it is a very small dataset, and how well the proposed learning algorithms will perform in more realistic scenarios requires further research. Furthermore, industrial control systems are attracting a range of adversaries, and the behaviour of these adversaries will be different to generated fault data. Thus, the anomaly detection systems suitable for fault detection and diagnosis on the TE process may not be suitable for detecting attacks. The security research literature in this area is in need of more studies on adversary modelling, and the creation of high quality public datasets. Whilst the Researchers from Kaspersky Lab made available their simulated attack data on the TE process model [123], these contain only a few types of attack. The SWaT dataset seems to be the only open-source data in this area.

3.2 Evolutionary Computation

Identifying those attacks that cause most damage against the components of a complex system with non-linear dependencies can be defined as a combinatorial optimisation problem. Developing attacks against existing defence mechanisms is an adversarial learning problem. Evolutionary computation provides promising solutions to these problems.

Due to the complexity of the problem, conventional methods such as exhaus-

tive search are not useful. Local search methods, such as *hill climbing*, which start from a single solution and move towards a local best by selecting a new solution in the neighbourhood are, by definition, local in scope. These solutions suffer from a major weakness that makes them unsuitable for most problems: success depends on the single initial starting point, and so they are prone to becoming stuck in local optima, especially for complex noisy search spaces. *Random search* algorithms are inefficient [148]. There are two popular search methods *tabu search* [149], a local search with memory; and *simulated annealing* [150], an iterative search imitating the annealing process in metallurgy. These algorithms may sometimes perform better than evolutionary approaches [151]; however, their application to problems in security domain is limited, and we are not aware of any studies using them for evolving attacks. As a result, we decided to eliminate them from consideration and to focus instead on how best to utilise evolutionary algorithms. This decision was supported by our own earlier work, [152] [153], which showed that evolutionary algorithms can be applied successfully to evolve attackers against an intrusion detection system protecting a wireless sensor network.

3.2.1 Evasion and Adversarial Learning

We assume an adversary who has control over her attack vector and knows the feature space used by the anomaly-based intrusion detection system. However, she does not know the detail of the underlying detection mechanisms. So far as she is concerned, the detection is a black-box that can be queried. She queries the detection with the attack vector, and obtains a detection probability. Her goal is to find attacks that cause damage whilst evading detection and using the least effort. Thus, the problem becomes a search problem with objectives.

Evolutionary algorithms [148] are inspired by analogies from natural selection and include elements such as mutation, selection and crossover. They are categorised as global search heuristics, which are used for finding solutions to optimisation and search problems. In the last two decades, evolutionary algorithms have been one of the most promising tools to emerge for solving challenging real-world problems with multiple conflicting objectives, chaotic disturbances, random-

ness, and complex non-linear dynamics that can be too complex for conventional algorithms to handle [154]. Evolutionary algorithms have been successfully applied to real-world problems, such as: electrical energy consumption forecasting [155]; improving weather prediction [156]; smart grid management [157]; understanding evolution of antimicrobial resistance [158]; optimising UAVs [159]; and optimisation of smart buildings [160], as well as a range of security problems. The most widely known evolutionary algorithms are genetic algorithms (GA). To solve a problem, GAs start with an initial population of randomly generated list of candidate solutions represented as chromosomes (also called the genotype or the genome). This evolves towards a population of better solutions over a number of generations using the Darwin's principle of natural selection, as implemented by genetic operators [161]. Chromosomes can be represented using a wide variety of data structures, including binary and real-numbers.

Some of the earlier work on evading intrusion detection was categorised under a concept called *mimicry attack*. The notion of mimicry attacks, and a theoretical framework for them, were introduced by Wagner et al. [162] with the aim of creating attacks that take steps to hide their existence from the IDS. They introduced the concept as a theoretical framework and called for further research in this area. The authors developed a framework to evolve a malicious sequence of system calls against the IDS with the intention of evading detection, and yet carrying out malicious behaviour. Similarly, Tan et al. [163] illustrated how adversaries can generate attacks that could render anomaly-based intrusion detection ineffective. The attacks they considered were those that exploited the privileges of UNIX systems programs, in the presence of an open-source anomaly detector (Stide). The attacks were generated using evolutionary algorithms by modifying malicious system call sequences so that they are ordered in a legitimate manner.

Evolutionary algorithms have been widely studied to improve the defence mechanisms used within IT networks. They are one answer to the problem of how to detect new forms of attack (exploiting unknown vulnerabilities) and to use this knowledge to improve detection. Early work in this area focused on generating new

attack signatures (rules) to detect new types of intrusions. Forrest et al. [164] [165] showed how to identify weaknesses in existing anomaly detection systems, and how one can manipulate these weaknesses to generate new attacks that are not detected by the existing IDS. Using the open-source DARPA Intrusion Detection dataset [166], Li [167] showed how to use genetic algorithms to evolve new rules that could help identify abnormal network behaviour before attacks take place. These rules are evolved by modifying network protocol fields. A significant number of studies followed this work, using GA to design better sets of rules for intrusion detection purposes, testing various attacks against benchmark network datasets [168] [169], [170] [171] [172] [173] [174]. Budynek et al. [175] used GA to evolve attack scripts against a simulated operating system, and covered their tracks to avoid detection from log file analysis.

Another evolutionary approach that has been widely used to evolve rules against intrusion detection is Genetic Programming (GP) [176]. GP extends the idea of GA to provide better expressive ability, by evolving more complex tree structures as opposed to lists in GA. Lu and Traore [177] produced one of the earliest works to apply GP to the detection of new forms of network intrusion. This work involved designing rules using Internet network parameters (e.g. protocol type, number of connections from the same source). They used the DARPA dataset [166] to show that GP can be used to evolve new attack rules to detect known or novel attacks on the network. Pastrana [178] used GP against a decision tree classifier (C4.5) and a Bayesian classifier (Naive Bayes). As before, GP evolves rules by making changes to network parameters.

In our previous work, [152] [153], GP was used to develop attacks, intended to help identify vulnerabilities in detection before they can be exploited by the attackers and so to improve the IDS. To assess the effectiveness of our approach, we tested against a wireless sensor network (WSN) with publish-subscribe communications, defended by an artificial immune intrusion detection system [179]. The use of GP for WSNs had not been studied in much detail prior to our work. The GP was used to evolve cache poisoning attacks against the WSN, and the evolved

attacks were successful in suppressing significantly number of legitimate messages while decreasing the likelihood of detection. Then, the IDS was improved by tuning the parameters based on the attack vectors. Kayacik et al. [180] used GP to evolve variants of buffer overflow attacks against a open-source (Snort) signature-based IDS. In [181], they used multi-objective genetic programming to generate multiple mimicry buffer overflow attacks to evade detection. In [182], the same researchers compared two attack generation approaches, "white-box" and "black-box", in which the white-box assumes internal knowledge of the detectors, and the "black-box" is limited by the response from the IDS.

Genetic algorithms have also been successfully applied to the evolution of malware samples designed to evade detection. Noreen et al. [183] used a genetic algorithm to evolve computer viruses against commercial antivirus software. Meng [184] used a multiobjective evolutionary algorithm (the Indicator-Based Evolutionary Algorithm [185]) to evolve Android malware against open-source malware detection tools. Calleja et al. [186] also evolved Android malware using genetic algorithms, but with the aim of forcing misclassification (i.e. if the target belongs to class A, evolve to classify it as another class) rather than evading the IDS. This was used against a malware detection system modelled as a decision tree classifier (C4.5) and 1-nearest neighbour algorithm. They successfully fooled the detection to misclassify 28 out of 29 cases. Aydogan and Sen used [187] GP to evolve Android malware and tested it against anti-virus systems available for mobile security. Xu et al. [188] attempted to evade malware classifiers for PDF files since PDF files are often used by adversaries to hide malware. The GP successfully evolved malware against two malware classifiers (a random forest classifier and a SVM classifier).

Genetic algorithms have also been applied to other defence mechanisms. The study by John et al. [189] applied genetic algorithms to the improvement of moving target defence, in which the attack surface is changed to disrupt the intelligence gathered by the attacker (often during the reconnaissance phase). Experiments were carried out on a prototype moving target defence system to evolve diverse set of security configurations based on old solutions to form new defence configurations.

In the study carried out by Dewri et al. [190], multiobjective optimisation was used with genetic algorithms to investigate optimal security measures for a system, defined using the attack tree model.

3.2.2 Coevolution Approaches

In biology, the term coevolution describes the process in which two or more species, e.g., predators and their prey, reciprocally cause changes in each other's evolution. The concept of coevolution has been incorporated into evolutionary computation [191] [192] [193] to solve increasingly complex problems. It has also been used to model the arms race between attackers and defenders, where both attackers and detection change over the time of the evolution. The studies discussed earlier relied on offline or manual changes to detection systems once new attacks were learned.

Rush et al. [194] simulated a coevolutionary agent-based network defence system, to coevolve attacker and defender strategies, in which the goal of the attacker is to explore and exploit the machines on a network, and the goal of the defender is to select the appropriate protection measure based on the detection and mitigation techniques (e.g. shutting down targetted machines based on a threshold). The authors carried out experiments using a simple simulation environment they implemented for this study; nevertheless, the work is a proof of concept that shows the potential applicability of coevolution to computer network security. Research carried out by Garcie et al. [195], applies coevolution to a peer-to-peer network in which attackers are tasked with disrupting the network by carrying out DoS attacks against a set of nodes for some duration, and the defender's goal is to protect the network by selecting the one of three routing strategies. The fitness of opposing strategies is defined as a multiobjective problem reflecting the goal of the attackers and defenders. The authors tested several coevolutionary techniques, and found that their adaption of the Incremental Pareto-Coevolution Archive (IPCA) algorithm [196] produced better results.

Coevolutionary concepts have also been investigated: to prevent faults and cascading blackouts in electric power transmission systems [197] [198]; to automate red teaming for military scenarios [199]; and to improve the performance of mal-

ware detectors [200].

3.3 Summary

In this chapter, we surveyed the literature relevant to our work. There is a large body of work related to intrusion detection for industrial control systems. However, the focus of these studies has largely been on the network level and, on their own, they are not sufficient to protect the security of ICS as they cannot detect the anomalous behaviour that arises if the data flow between sensors, controller and actuators is comprised. Furthermore, these studies, have also limitations due to oversimplified threat modelling, and lack of realistic test environments and datasets. Somewhat surprisingly, few studies focus on the interactions between different network layers of industrial control systems. Most of the studies relevant to our work are carried out by the fault detection and diagnosis community, as there is a considerable literature focussed on fault detection for nonlinear systems. However, the applicability of their approaches to attack detection is not well understood, because the behaviour of adversaries is different to those generated by equipment malfunctions, signal interruptions or human error.

The security community is starting to pay some attention to the detection of attacks in the industrial control processes, which is encouraging. The focus of these studies has been on unsupervised deep neural networks, and we will explore some of these techniques in this thesis. However, due to shortage of open-source datasets, attack detection in process control is an area not well studied by the community.

In this chapter, we also covered the related work in evolutionary computation. Studies based on co-evolution, mimicry attacks and evasion techniques are all intended to help identify vulnerabilities before systems are attacked. These studies appear to be promising; however, with the exception of studies on evolving malware, they tend to focus on simple use cases and weak detection mechanisms. In this thesis, we will investigate if it is possible to use techniques from evolutionary computation to evolve attacks on a complex real-world system.

Chapter 4

Case Study, Threat Model and Methods

We based an investigation of research questions outlined in Section 1.2 on a realistic process model that represents a real-world ICS process, and a simulation environment in which we can study the security properties of and damage caused to a industrial plant. Setting up a laboratory based system, or even a hardware-in-the-loop (HIL) systems that combines with real devices was beyond scope of this thesis. Although there is some work related to ICS testbeds, this often fails to reproduce the process accurately [201]. In either case, simulations is essential to allow machine learning to conduct many experiments in a short space of time.

We focused on software simulations that are realistic, the most mature of which was the Tennessee Eastman process control problem [20]. This simulated system has credibility because it represents a real industrial process and was created and validated by the owners of that process. Our reasons for selecting the TE process model are: i) it is a well-known model that has been widely studied; ii) it is a complex, highly non-linear system with a number of components that reflects a real process; iii) safety and economic viability can be quantified; iv) the code and model is available, and have been revised and validated over the years; v) it continues to be a relevant model for both control and, more recently, the security community. The National Institute of Standards and Technology (NIST) has developed a hardware-in-the-loop cybersecurity performance testbed using the TE process model [202].

We are not aware of any other open source model that has these properties.

In this chapter, the TE process control model is introduced. The objective of the plant, the components, in terms of sensors and actuators, and the potential disturbances are illustrated. We introduce the threat model and discuss some attacks we will be investigating using the TE model. Next, we introduce the background material for evolutionary multiobjective optimisation, and explain the evolutionary algorithms we have selected for optimisation. Finally, we give a brief overview of machine learning and deep learning, and introduce the techniques we identified as potential methods for detection.

4.1 Case Study: Tennessee Eastman Process Control Problem

The Tennessee Eastman (TE) process control problem was first introduced at a meeting organised by the American Institute of Chemical Engineers AIChE in 1990 by Downs and Vogel [20]. The purpose of this problem was to provide the academic process control community with a realistic model of a real-plant wide industrial process. The behaviour of the plant was provided in Fortran code, with a flow-sheet, and a description of the process. Since the 1990s, the TE process model has been studied by many people in the control community: for control design [203, 204, 205, 206] optimisation, fault detection and diagnosis; [129, 134]; for teaching; and now, by security researchers, to study the security of ICS [119, 207, 208, 209]. The TE model is based on a real process; however, some changes were made to keep the identity of the reactants and products secret. The process has a total of eight components: four gaseous reactants (A, C, D, E), two products (G, H), an inert component (B) and a by-product (F). The reactions are [20]:

A(g) + C(g) + D (g) $\rightarrow G(liq)$, Product 1,

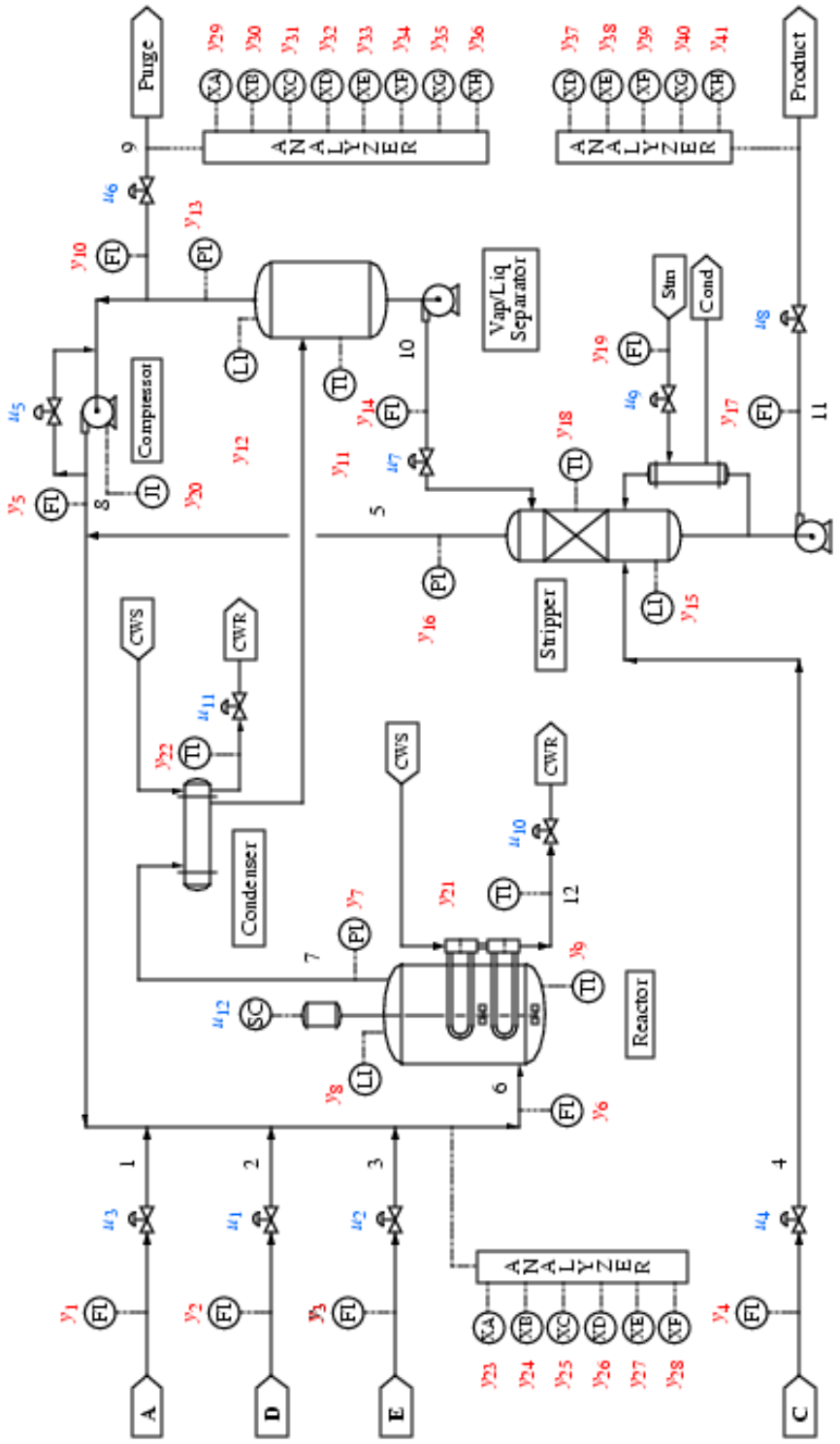
A(g) + C (g) + E (g) $\rightarrow H(liq)$, Product 2,

A(g) + E (g) $\rightarrow$F(liq), Byproduct,

3D(g) $\rightarrow 2F(liq)$, Byproduct.

The reactions are irreversible and exothermic, and their rates are a function of temperature, as determined by an Arrhenius equation [20]. The process is illustrated in Figure 4.1, and consists of five major components [20]: the reactor, the product condenser, a vapour-liquid separator, a recycle compressor and a product stripper. The gaseous reactants (A, C, D and E), are fed to the reactor, in which they react to form liquid products. The process has four feed streams (A, D, E and A+C), one product stream (a mix of G and H) and one purge stream. Most of the inert component (B) enters as part of C feed. The gas phase reactions are catalysed by a non-volatile catalyst dissolved in the liquid phase within the reactor. The reactor uses an internal cooling capability for removing the heat produced by the reaction. The products leave the reactor as vapours along with the unreacted feeds, while the catalyst remains in the reactor [20]. The product stream from the reactor goes through a cooler that condenses the products and, from there, it passes to a vapour-liquid separator. Noncondensed components are recycled back to the reactor feed through a centrifugal compressor. Condensed components are moved to product stripper column to strip any remaining reactants with feed stream 4. Products (G and H) come out from the bottom of the stripper and they are separated in a downstream refining section that is not present in this model [20]. The inert component and byproducts are purged from the plant as vapour through the vapour-liquid separator. The process has six modes of operation with different G/H mass ratios designed to produce different production outputs at different rates to meet market demands.

There are 41 process variable measurements known as XMEAS (sensors illustrated in Table 4.1) and 12 manipulated variables (known as XMV (valves/actuators, illustrated in Table 4.2) that are involved in controlling and monitoring the plant.



PI: Proportional-integral, **FI:** Flow Indicatior, **TI:** Temperature Indicator, **LI:** Level Indicator

Figure 4.1: Tennessee Eastman process control problem [20] [210]

Measurements	Variable Number	Variable Name
Continuous	XMEAS (1)	A feed (stream 1)
	XMEAS (2)	D feed (stream 2)
	XMEAS (3)	E feed (stream 3)
	XMEAS (4)	A and C feed (stream 4)
	XMEAS (5)	Recycle flow (stream 8)
	XMEAS (6)	Reactor feed rate (stream 6)
	XMEAS (7)	Reactor pressure
	XMEAS (8)	Reactor level
	XMEAS (9)	Reactor temperature
	XMEAS (10)	Purge rate (stream 9)
	XMEAS (11)	Product separator temperature
	XMEAS (12)	Product separator level
	XMEAS (13)	Product separator pressure
	XMEAS (14)	Product separator underflow (stream 10)
	XMEAS (15)	Stripper level
	XMEAS (16)	Stripper pressure
	XMEAS (17)	Stripper underflow (stream 11) XMEAS
	XMEAS (18)	Stripper temperature
	XMEAS (19)	Stripper steam flow
	XMEAS (20)	Compressor work
	XMEAS (21)	Reactor cooling water outlet temperature
	XMEAS (22)	Separator cooling water outlet temperature
Reactor Feed Analysis (Sampled)	XMEAS (23)	A
	XMEAS (24)	B
	XMEAS (25)	C
	XMEAS (26)	D
	XMEAS (27)	E
	XMEAS (28)	F
Purge Gas Analysis (Sampled)	XMEAS (29)	A
	XMEAS (30)	B
	XMEAS (31)	C
	XMEAS (32)	D
	XMEAS (33)	E
	XMEAS (34)	F
	XMEAS (35)	G
	XMEAS (36)	H
Product Analysis (Sampled)	XMEAS (37)	D
	XMEAS (38)	E
	XMEAS (39)	F
	XMEAS (40)	G
	XMEAS (41)	H

Table 4.1: Process variable measurements of the TE process [20]

Variable Number	Variable Name
XMV(1)	D Feed Rate (stream 2)
XMV(2)	E Feed Flow (stream 3)
XMV(3)	A Feed Flow (stream 1)
XMV(4)	C Feed Rate (stream 4)
XMV(5)	Compressor Recycle Valve
XMV(6)	Purge Valve (stream 9)
XMV(7)	Separator Pot Liquid Flow (stream 10)
XMV(8)	Stripper Liquid Product Flow (stream 11)
XMV(9)	Stripper Steam Valve
XMV(10)	Reactor Cooling Water Flow
XMV(11)	Condenser Cooling Water Flow (stream 13)
XMV(12)	Agitator Speed

Table 4.2: Process manipulated variables of the TE process [20]

The first 22 XMEAS variables are continuous, and the remaining process variables are sampled with frequencies that vary between 0.1-0.25 hours, and a dead time (the time between the sample taken and the analysis is complete) also of 0.1-0.25 hours [20]. The main control objectives of the plant are [20]: to maintain the process variables at the desired values; to ensure that the operational conditions are within the equipment constraints; reduce the variability of the product rate and product quality during disturbances; minimise valve movement; and to recover fast and smoothly from disturbances, production and product changes. For example, product composition variability of greater than $\pm 5 mol\%$ G is assumed to be harmful, and therefore plant needs to avoid this. The process operating constraints are illustrated in Table 4.3. A detailed description of the TE process model can be found in [20].

Process Variable	Normal Operating Limits		Shut Down Limits	
	Low Limit	High Limit	Low Limit	High Limit
Reactor Pressure	none	2895 kPa	none	3000 kPa
Reactor Level	50% ($11.8\ m^3$)	100% ($21.3\ m^3$)	$2.0\ m^3$	$24.0\ m^3$
Reactor Temperature	none	150°C	none	175°C
Product separator Level	30% ($3.3\ m^3$)	100% ($9.0\ m^3$)	$1.0\ m^3$	$12.0\ m^3$
Stripper Base Level	30% ($3.5 m^3$)	100% ($6.6 m^3$)	$1.0\ m^3$	$8.0\ m^3$

Table 4.3: TE process operating constraints [20]

The TE process problem makes no recommendation as to what need to be controlled, and leaves the selection of controlled variables and control strategies to the control engineers. Most proposed solutions do not control all the variables. The process model used in this work is developed by Ricker, and it is available from his home page [124]. The code is implemented in C, with a MATLAB/Simulink interface available via an S-function implementation. The control control strategy used is by Larsson et al. in [210]. Isakov and Krotofil [211] extended the Simulink model by enhancing it with Simulink blocks that enable one to carry out integrity and DoS attacks on the process measurement and manipulated variables. We extended their model to add replay attacks, have more control over the disturbances (which can now be turned on at a particular time) and corrected some small problems we saw in their implementation. The attacks in this report are carried out using the model we extended. The simulated control system has 18 proportional-integral controllers, 16 process vector measurements XMEAS (1-5,7-12,14-15,17,31,40), and 9 setpoints controlled by eight multivariable control loops and one single feedback control loop [208] [210]. The mode used throughout the experiments in this thesis is Mode 1 with a given 50/50 mass ratio between components G and H [20].

The problem includes 20 disturbances (defined as faults by the fault detection and diagnosis studies) illustrated in Table 4.4. These can be turned on and off.

4.1.1 Disturbances

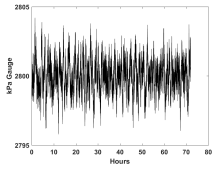
One of the objectives of processes like TE is to react appropriately to the disturbances that occur as the results of random fluctuations in the real-world. Thus, the control configurations need to recover quickly and smoothly from the variations that are caused by process dynamics. Table 4.4 illustrates typical disturbances of the process, using the acronym from the original paper, IDV for Disturbance. To understand the full effect of the disturbances, the model was executed using the disturbances in Table 4.4. Figure 4.2 illustrates the variability in reactor pressure when disturbances are turned on. Disturbances 6 and 8 are harder to handle. Disturbance 8 (IDV 8) introduces random variation in A, B and C feed flow composition (stream 4), and generates large variations in reactor pressure (Figure 4.2) and the flow rate of

Variable Number	Process variable	Type
IDV (1)	A/C feed ratio, B composition constant(stream 4)	Step
IDV (2)	B composition, A/C ratio constant (stream 4)	Step
IDV (3)	D feed temperature (stream 2)	Step
IDV (4)	Reactor cooling water inlet temperature	Step
IDV (5)	Condenser cooling water inlet temperature	Step
IDV (6)	A feed loss (stream 1)	Step
IDV (7)	C header pressure loss-reduced availability (stream 4)	Step
IDV (8)	A, B, C feed composition (stream 4)	Random Variation
IDV (9)	D feed temperature (stream 2) Random	Variation
IDV (10)	C feed temperature (stream 4)	Random Variation
IDV (11)	Reactor cooling water inlet temperature	Random Variation
IDV (12)	Condenser cooling water inlet temperature	Random Variation
IDV (13)	Reaction kinetics	Slow Drift
IDV (14)	Reactor cooling water valve	Sticking
IDV (15)	Condenser cooling water valve	Sticking
IDV (16)	Unknown	Unknown
IDV (17)	Unknown	Unknown
IDV (18)	Unknown	Unknown
IDV (19)	Unknown	Unknown
IDV (20)	Unknown	Unknown

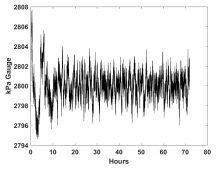
Table 4.4: TE process disturbances [20]

the products. The process is able to control and reject IDV 8 disturbances. However, as shown in Figure 4.2, the IDV 6 disturbances are the most drastic: the process is unable to cope after 7 hours. IDV 6 disturbs the stoichiometric ratio of components A and C and drives the concentration of the purge gas stream down rapidly.

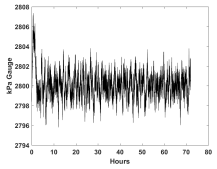
Fault detection and diagnosis studies use these disturbances to distinguish between normal operating conditions and disturbances.



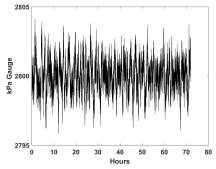
Normal 1



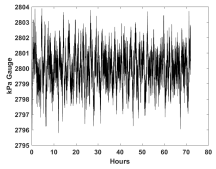
Disturbances IDV 1



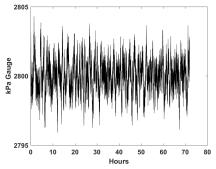
Disturbances IDV 2



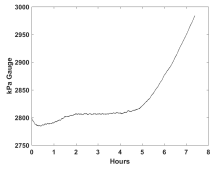
Disturbances IDV 3



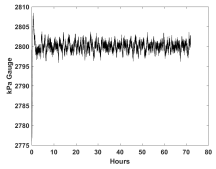
Disturbances IDV 4



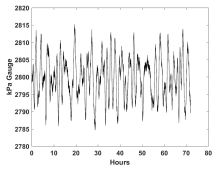
Disturbances IDV 5



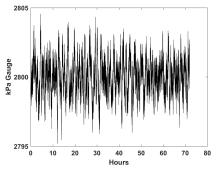
Disturbances IDV 6



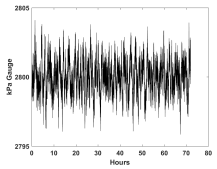
Disturbances IDV 7



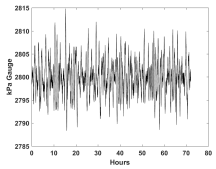
Disturbances IDV 8



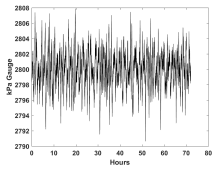
Disturbances IDV 9



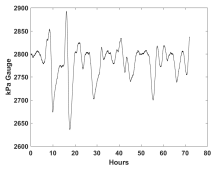
Disturbances IDV 10



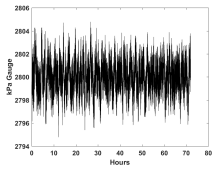
Disturbances IDV 11



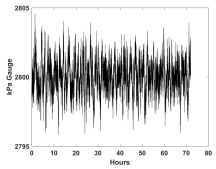
Disturbances IDV 12



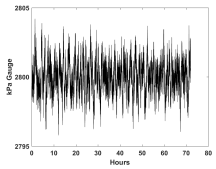
Disturbances IDV 13



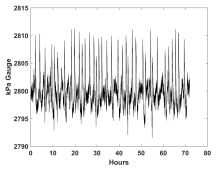
Disturbances IDV 14



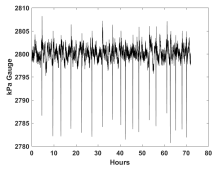
Disturbances IDV 15



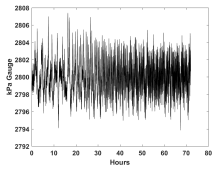
Disturbances IDV 16



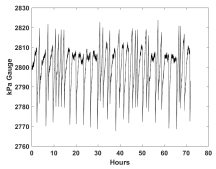
Disturbances IDV 17



Disturbances IDV 18



Disturbances IDV 19



Disturbances IDV 20

Figure 4.2: Behaviour of the TE process disturbances

4.2 Threat Model

To explore our research questions we analysed a number of attack scenarios against the TE model. We assume an adversary that has no special knowledge of the plant but is able to monitor and modify the measurements from process variables that are sent from sensors to the controller, and control commands that are sent from the control to actuators [212]. As past attacks show, these are common attack vectors that are often used to carry out attacks. How attackers compromised these signals is outside the scope of our work. However, it is worth noticing that there are examples of situations in which networks are compromised. As discussed in Chapter 2, the security of ICS and, in particular, the communication protocols is poor; it is regrettably a low barrier to attack in many existing systems. Our assumptions is that the adversaries can indeed obtain the necessary means to compromise the confidentiality of the system and so they have access to the communication between the control components. As discussed in Section 2.3.6, there are a wide range of threat actors, and their motives for carrying out the attacks are not all the same. For a process like the TE model, this could mean targeting the safety of the system, increasing the operating cost, and targeting the production quality:

- **Safety:** As illustrated in Table 4.3, the TE process has a set of operating constraints that are captured as normal plant operating limits and plant shutdown operating limits. The process should operate within the normal limits. If the process strays outside the shutdown limits, it will shut the plant down immediately [210]. These limits are put in place to protect the personnel, equipment and production, and to meet regulatory compliance requirements. For the TE model, when the process reaches the low or high limits of reactor pressure, reactor level, reactor temperature, product separator level or stripper base level, it will shut down. An adversary may therefore target the availability of the plant by attempting to it shut down.

- **Operational Cost:** The operational costs of the TE process are calculated according to the following equation [20]:

$$totalcosts = (purgecosts)(purgerate) + (productstreamcosts)(productrate) + (compressorcosts)(compressorwork) + (steamcosts)(steamrate) \quad (4.1)$$

An adversary may try to increase the total operating costs of the plant. This cost is based on the loss of raw materials, calculated considering the loss of product in the purge stream, product leaving in the product stream, the cost of associated with amount of compound F formed, and the cost of the compressor work and stripper steam costs [20].

- **Quality of the Product:** Chemicals are often classified into three grades: *reagent* (highest purity available), *laboratory* (relatively high purity) and *technical* (contains impurities), and prices are determined according to the purity. Manufacturing pure chemicals is an expensive process, and impurities not only determine the sale price for the products, but they may also cause harmful effects. An adversary may try to cause economic damage by attacking the quality of the product. Thus, an adversary may try to damage the process by causing composition variability.

Figure 4.3 shows our underlying threat model that is based on common attacks against the NCS. The adversary is capable of intercepting the communication from the sensor to controller, and controller to actuator. The attacks we consider are categorised as DoS, integrity (man-in-the-middle) and replay attacks. We will investigate the impact of these attacks in terms of measuring the impact on safety, operational cost, and product quality. In the following section, we briefly discuss what this means, and explain how the attacks are modelled.

4.2.1 Denial of Service Attacks

An adversary may attack the signal sent from the sensor to controller and or from controller to the actuator. In the past, DoS attacks were carried out on industrial control equipment by exploiting a range of vulnerabilities, including unauthenticated

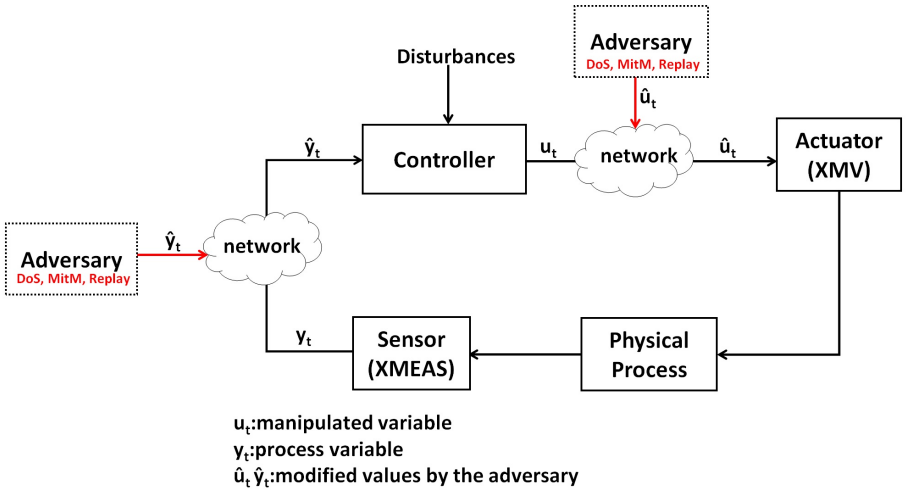


Figure 4.3: ICS attack model against the networked control system

remote attackers causing devices go into states such as defect mode [213] [214]; flooding the network with large amount of traffic to prevent legitimate packets from reaching the master devices or any other connected devices; sending incorrect data, for example longer parameters [215]; buffer overflows [216]; sending specially created packets to certain ports [217] [218]; and hardware vulnerabilities [219]. Often, these types of vulnerabilities require devices to be manually reset and/or patched to allow the system to continue with normal operation.

4.2.2 Integrity Attacks

This is a form of man-in-the-middle attack, in which the adversary modifies the process values that are sent from the sensors to the controllers, and/or from the controller to the actuators. This kind of attack can be carried out, firstly by exploiting the vulnerabilities that bypass authentication and escalate privileges using a variety of techniques such as scanning and sniffing the network [220]; weak architecture design that permits interception and change of authentication requests [221]; weak default passwords and insecure authentication generation [222]; missing authentication when exceptions occur (e.g. entering safe mode without security) [223]; security practices such as hard coded private SSH and HTTPS keys [224] and storing credentials without sufficient encryption [225]; using broken or insufficient cryptographic algorithms; lack of two-way (mutual) authentication; transmitting or

storing credentials in clear; keeping default credentials; inadequate authentication and integrity verification methods; poor access control mechanisms; back-doors intended for software update and maintenance; and exploiting insiders. The attack vector is not limited to this, and is constantly expanding.

4.2.3 Replay Attacks

An adversary may gather data from a sensor or controller for some duration, and then replay this data at another time to carry out a more subtle attack. It can be carried out using the same exploits as integrity attacks, but, in addition, needs a memory in which to record data sequences.

4.2.4 Attack Model

Others have investigated the impact of DoS and man-in-the-middle attacks on the TE model in terms of safety and economic consequences [226] and [208]. We are following their attack model; however, the focus of the previous study in [226] concentrated on a simplified version of the TE model, and [208] examined at single DoS and integrity attacks. We extend their analysis by undertaking a more comprehensive search and examining the possibility of forming combinations of attacks. The attack parameters are as follows:

- **Duration of the Attack**: The maximum length of the attack is limited to the process run time. In the used TE model, this is limited to 72 hours. An attack begins at time t_s and ends at t_e. It can start any time, between start of the plant and the end: $t \in [0-72]$. Let I_a be the attack interval, let y_i be the output of sensor i at time t, and let u_i be the controller to actuator signal i at time t. The modified signals $y_i^a(t)$, $u_i^a(t)$ are as follows:

$$y_i^a(t) = \begin{cases} y_i^{(t)}, for\, t \notin I_a \\ \hat{y}_i^{(t)}, for\, t \in I_a \end{cases} \tag{4.2}$$

$$u_i^a(t) = \begin{cases} u_i^{(t)}, for\, t \notin I_a \\ \hat{u}_i^{(t)}, for\, t \in I_a \end{cases} \tag{4.3}$$

where $\hat{y}_i^{(t)}$ and $\hat{u}_i^{(t)}$ are the modified values the adversary sends.

- **Attack Mode**: There are two types of attack modes built into the simulation [211]: interval and periodic. Interval attacks start from the time of the attack and continue for the given duration. The periodic attack starts from the starting time of the attack, and lasts until the duration of the attack is complete, but it does this periodically, as defined by setting a pulse (wait) period.

- **Attack Value**: For the *DoS attack*, we assume the response strategy for the controller or the actuator is to use the last received value as the current reading:

$$\begin{aligned} \hat{y}_i^{(t)} &= y_i(t_{s-1}) \\ \hat{u}_i^{(t)} &= u_i(t_{s-1}) \end{aligned} \tag{4.4}$$

where the t_s is the attack start time.

Man-in-the-middle attacks on systems similar to TE models involve forging the signals that go from the sensor to the controller and or from the controller to the actuator. The receiving end, for example the controller, will respond to the forged signal. Rather than sending random values, an attacker may prefer to listen to the transmitted signals and modify values so that they are still within the ranges of possible plant signals: this has the potential to cause damage. One way to achieve this is by modifying the sensor measurements (y_i) and manipulated values (u_i) using observed upper (maximum) and lower (minimum) limits [226]. For the remaining of the thesis, we will call these attacks IntegrityMin and IntegrityMax attacks.

The IntegrityMin is where the actual output of the sensor or controller signal i at time t is replaced with a minimum value:

$$\begin{aligned} \hat{y}_i^{(t)} &= \min_{t \in T}(y_i(t)) \\ \hat{u}_i^{(t)} &= \min_{t \in T}(u_i(t)) \end{aligned} \tag{4.5}$$

The IntegrityMax is where the actual output of the sensor or controller signal

i at time t is replaced with a maximum value:

$$\begin{aligned} \hat{y}_i^{(t)} &= \max_{t \in T}(y_i(t)) \\ \hat{u}_i^{(t)} &= \max_{t \in T}(u_i(t)) \end{aligned} \tag{4.6}$$

The replay attack manipulates signals that are sent from the sensor to controller, or from controller to the actuator as in the integrity attack but, this time, it repeatedly replays the signals it collected earlier for the duration of the attack:

$$\begin{aligned} y_i^r &= [y_i^{(r_{start})}, ..., y_i^{(r_{end})}] \\ u_i^r &= [u_i^{(r_{start})}, ..., u_i^{(r_{end})}] \\ \hat{y}_i^{(t)} &= \hat{y}_i^r[t \bmod \text{len } y_i^r] \\ \hat{u}_i^{(t)} &= \hat{u}_i^r[t \bmod \text{len } u_i^r] \end{aligned} \tag{4.7}$$

where y_i^r and u_i^r are the signals recorded by the adversary from the replay period, r_{start} to r_{end}.

- **Attack Targets**: Possible attack targets are process variable measurements sent by the sensors or manipulated variables received by the actuators. The control strategy selected for the TE process uses 16 process variable measurements (XMEASs) from sensors, and 12 manipulated values (XMVs) for valves. XMV 5, XMV 9 and XMV 12 are fixed, and so they are not used by the control strategy. We denote the remaining XMEAS and XMV as the potential targets for attacks. Controllers can be attacked by manipulating the setpoints but, given the size of the search space and limited resources, we will focus on the sensors and actuators.

4.3 Evolutionary Algorithms and Multiobjective Optimisation

In this section, we introduce the terminology that relates to the evolutionary algorithms, and explain the details of two candidate methods we have selected for further investigation: the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [227] and Strength Pareto Evolutionary Algorithm 2 (SPEA2) [228].

Genetic algorithms (GA) [148], like other members of the evolutionary computation family, are inspired by genetic processes of biological organisms, mimicking the process of natural evolution. Although these techniques are similar, they differ in the way they use genetic operators and how they are represented (the data structures used). To cover the new developments in the field, many people have been using general umbrella terms such as evolutionary computing or evolutionary algorithms. In this thesis, we use the term *evolutionary multiobjective optimisation (EMO)* to describe genetic algorithms for multiobjective optimisation. Furthermore, to avoid any confusion, the evolutionary algorithms developed as part of the work reported in this thesis implements the evolutionary concepts shown in Figure 4.4.

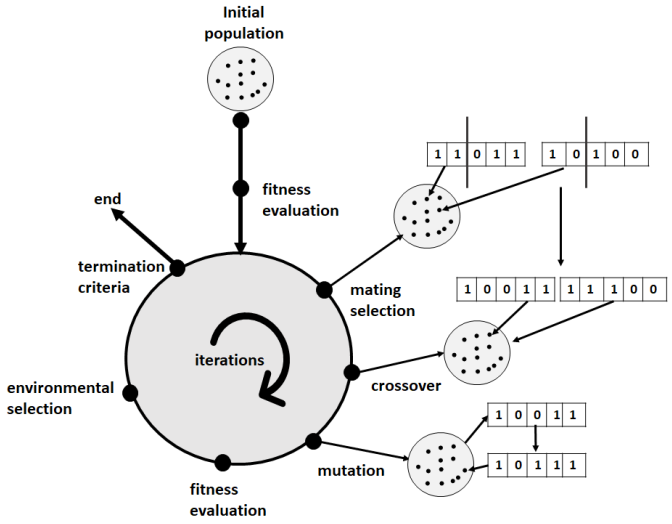


Figure 4.4: General process of an evolutionary algorithm [229]

The terms used in discussing genetic algorithms are generally accepted and used by other evolutionary algorithms. A GA starts with a *population* of *individuals*,

each describing a possible solution to the given problem. Each individual in the population gets assigned a *fitness* value according to a *fitness function* that calculates how close the individual is to the desired solution. Before, a GA can be executed, a suitable *representation* or (*encoding*) and a *fitness function* need to be devised to evaluate the performance of the individuals. These solutions are represented as a set of parameters. These parameters are known as *genes*. The individual solutions composed of genes are *chromosomes*. Each gene may have some number of values called *allele*, and the position of the gene is known as its *locus*. The term that is often used to describe the genetic matter of the individuals in decision space (i.e. referring to the encoding of the individual, chromosome) is *genotype*. The performance of this genotype can be evaluated once it is expressed in the objective space known as the *phenotype*. The individuals in the population are subject to reproduction and evaluation for a number of iterations, known as *generations*, with the aim that individuals evolve in a way that increases their fitness.

During reproduction, individuals are selected from the current population and *variation* is performed to change the genetic material of the individuals. This variation involves *crossover* (also called *recombination* and *mutation*). Given two individuals in the current population (*parents*), crossover operators (e.g. one-point crossover, two-point crossover, ordered crossover, uniform crossover) are used to randomly select some portions of the parents to generate a new *offspring*, inheriting some genes from each parent. The number of individuals selected for crossover is defined as a probability, typically a rate of 0.8 or greater is selected to ensure the parents have a high chance of passing their genes to their offspring. On the other hand, mutation (e.g. bit flip mutation, swap mutation, scramble mutation, uniform mutation) involves making a small alteration in a current individual, and the probability of the mutation is often small, typically in the region of 0.1, to avoid delaying convergence unnecessarily. These parameters are often selected using self-tuning techniques. If the rates of crossover and mutation do not add up to 1.0, then some individuals are copied, *reproduced* (verbatim) to the next generation.

As with most evolutionary algorithms, *selection* is used to ensure the individ-

uals with the highest fitness value survive to the next generation. This is known as *environmental selection*. The selection of individuals that are used to produce offspring is known as *mating selection*.

If the GA is effective, it will evolve the population over successive generations such that the fitness of the solutions increases in each generation, towards the global optimum. A GA *converges* when it progresses to the point at which most of the individuals in the population are identical and diversity is at a minimum. An undesirable, but common problem in GAs is *premature convergence* [230] when a few individuals with high, but not optimal, fitness values rapidly dominate the population, causing it to get stuck at a local minimum (or maximum). When this occurs, the GA loses its ability to improve the fitness of the individuals in the population, as the crossover of similar parents fails to generate offspring that are superior to their parents.

4.3.1 Multiobjective Optimisation

In many real-world problems, decisions need to be made based on multiple competing or conflicting objectives and constraints. This is often the case in security for making decisions related to cyber security investment or security hardening, where resources must be allocated on the basis of risks, in the presence of constraints such as a limited security budget for buying control measures. In such situations, where there might not be a single solution, formulating the problem as a multiobjective optimisation (MOO) with multiple choices can help to balance the trade-offs among the objectives in a more effective manner [190, 231, 232]. These approaches search for the set of non-dominated solutions (i.e. *individuals* in evolutionary computation). A solution is defined as being non-dominated if there are no other solutions that would improve any objective without degrading one or more of the other objectives [233]. Once the set of these solutions is identified, known as the *non-dominated set* or the *Pareto-optimal set*, then the decision maker can make decisions by examining these solutions.

MOO takes problems with multiple objectives and simultaneously seeks to optimise all objectives and provide solutions on, or close to, the Pareto-optimal set.

More often, this is an estimate because real-world problems are too complex to allow the complete Pareto-optimal set to be determined, either because the search space is too large or because obtaining solutions is costly in time and computation.

The multiobjective optimisation problems we have are related to increasing the impact of an attack whilst avoiding detection. We are concerned with finding optimal solutions for the following cases:

1. **Attack the safety of the plant**: Minimise plant operating time, minimise effort required to carry out attacks

2. **Cause economic damage:** Maximise operating cost, minimise effort required to carry out attacks

3. **Cause economic damage and avoid detection:** Increase operating cost, minimise Detection Rate, minimise effort

Such real world problems are often complex and NP-hard, many containing parameters related to decision making, objectives and constraints. Generating attacks against a process like the TE process could involve the selection of: attack target (controllers, sensors, actuators); attack types; parameters for these attacks; attack start times; attack duration; and so forth. As explained earlier, in practice, identifying the exact Pareto-optimal set for such problems is often not feasible. In these cases, the concept of estimation, ε-dominance [234] and ε-approximation are applied. Evolutionary algorithms have become a promising approach for solving multiobjective optimisation problems [235] that try to find near optimal solutions using reasonable computational power due to their inherent parallelism, and their ability of generating a set of solutions in a single run [236].

4.3.1.1 Multiobjective Optimisation Terminology

Without loss of generality, we assume that a MOO problem involves minimising all objectives. A MOO minimisation problem is defined in terms of a set of m decision variables and n objectives [233]:

$$Minimise : y \rightarrow f(x) = (f_1(x), f_2(x), ..., f_n(x)) \tag{4.8}$$

where individuals from the decision space **X** are mapped into the objective **Y** space:

$$\begin{aligned} \mathbf{x} &= (x_1, x_2, ..., x_m) \in X \\ \mathbf{y} &= (y_1, y_2, ..., y_n) \in Y \end{aligned} \tag{4.9}$$

and where **x** is the decision vector and **y** is the objective vector. A decision vector $\mathbf{u} \in X$ **dominates** another one $\mathbf{v} \in X$ ($\mathbf{u} < \mathbf{v}$) if and only if (iff) [233]:

$$\forall_i \in (1,2,...,n): \quad f_i(\mathbf{u}) \leq f_i(\mathbf{v}) \quad \text{and} \quad \exists_i \in (1,2,...,n): \quad f_j(\mathbf{u}) < f_j(\mathbf{v}) \tag{4.10}$$

That is, none of the objectives in **u** is worse than the objectives in **v**, and **u** contains at least one objective that is strictly better than **v**.

Based on these relations, the non-dominated individuals can be defined. Let $x \in F$ be an arbitrary decision vector. x is non-dominated regarding the set $F' \subseteq F$ iff there is no vector in F' which dominates x [233]:

$$\nexists \mathbf{x}' \in F' : \mathbf{x}' < \mathbf{x} \tag{4.11}$$

Based on these definitions, the decision vector **x** is *Pareto optimal* iff **x** is defined as non-dominated regarding F. The *Pareto optimal set* is defined as the set of all non-dominated individuals, that is Pareto optimal individuals. The corresponding objective values of the Pareto optimal set is known as *Pareto optimal front* (also known as Pareto frontier)[233].

4.3.2 Evolutionary Multiobjective Optimisation Algorithms

Application of evolutionary algorithms in multiobjective optimisation, known as evolutionary multiobjective optimisation (EMO) [233], has attracted research interest from researchers with a wide range of backgrounds, and has become an active research areas in the field of evolutionary computation. As a result, a number of algorithms have been proposed to solve multiobjective optimisation problems. In general, EMO algorithms (EMOA) can be classified into three groups based on their selection operators: Pareto-based, aggregation-based and indicator-based al-

gorithms [237]. The Pareto-based algorithms first measure the quality of individuals based on Pareto dominance, and then a secondary operation is used to make selections among the non-dominated individuals to maintain population diversity. Some of the more popular Pareto-based EMOA include the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [227], Strength Pareto Evolutionary Algorithm 2 (SPEA2) [228], Pareto archived evolution strategy (PESA) [238], Pareto Envelope-based Selection Algorithm II (PESA-II) [239] and Niched Pareto Genetic Algorithm for Multi-Objective Optimization (NPGA) [240].

Non-Pareto algorithms measure the quality of individuals based on aggregation or indicator. Aggregation-based algorithms reformulate the MOO problem into a set of scalar optimisation sub-problems, optimise them simultaneously and aggregate the objectives using an aggregation function to accumulate a single scalar value [237]. Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D) [241] is the most popular aggregation-based EMOA. Indicator-based EMOA use a performance indicator function (e.g. s-metric or hypervolume, see Section 6.2) to determine the quality of the solution sets. S-metric Selection-EMOA (SMS-EMOA) [242], indicator-based EA (IBEA) [185] and HypE [243] are the three of the best-known indicator-based EMOA.

Due to the computationally intensive nature of this work, and the selected TE model simulator in MATLAB being slow, it was not possible for us to compare a wide range of EMO algorithms. We decided to focus on the Pareto-based algorithms, among which NSGA-II and SPEA2 are well-known, and generally accepted as being two of the most reliable forms of EMOA. They have both been subject to changes over the years to improve performance and accuracy, and they have been successfully applied to a range of problems. They work well for problems with a small number of objectives. They are also straightforward to implement and they can easily be altered to meet the needs of the problem. Considering these options, we decided to utilise NSGA-II and SPEA2 in our work. In the following sections, we present how these two algorithms work and in subsequent chapters, we explain how we have utilised them.

4.3.2.1 The Non-Dominated Sorting Genetic Algorithm II (NSGA-II)

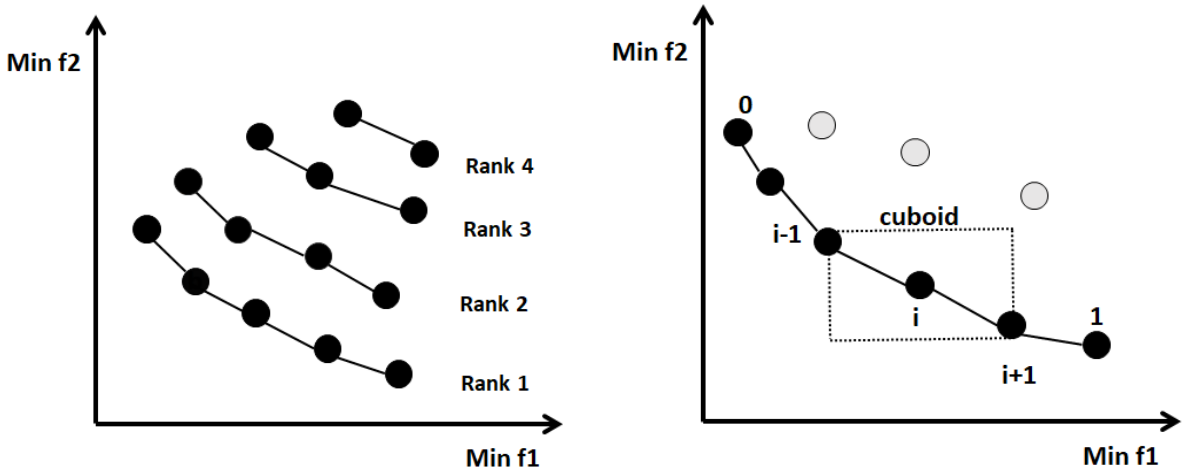
The Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [227] [244] is one of the most widely used EMOAs. It is an improved version of the earlier NSGA [245] and is computationally faster, better at keeping good individuals once they are found through use of elitism, and better at preserving diversity through the guidance of the crowding distance operator. Before describing the whole process, we first explain two important components of NSGA-II: fast non-dominated sorting and crowding distance computation. The non-dominated sorting procedure takes a population P_0, containing N individuals, applies a ranking to each individual in P, and returns a list of non-dominated fronts F using the steps illustrated in Algorithm 1. For each individual p in a population P, a comparison is made to all other individuals q in P. If p dominates q then q is added to the set of individuals dominated by p, S_p, however, if p is dominated by q then the domination counter of p, n_p is increased by 1 (i.e. sum of individuals that dominate p) [244]. If there are no individuals that dominate p, then p belongs to the first front, F_1 [244]. While F_i is not empty, for each member p in F_i and q in S_p, if the size of n_q is 0, then q is added to the next front [244]. This process is used to identify all fronts, each assigned a ranking. Assuming it is a two-objective minimisation of fitness, Figure 4.5a illustrates an example of the ranking, and assignment of individuals to these fronts. Each individual is assigned a rank according to its non-domination level, where 1 is the best level, 2 is the second best level, and so on [244].

In the next step, NSGA-II computes the crowding distance of each individual within its front to determine the density of individuals surrounding a particular individual in the population [244]. NSGA-II considers each of the values of the M objectives independently, and sums the distances between their neighbouring individuals. As we show later, this density value is used to select individuals within a set. Figure 4.5b illustrates an example of the crowding distance of an individual, i, within its front with two objectives $M=(f1, f2)$. Each of these objectives are processed independently and nearest neighbour individuals change for each objec-

Algorithm 1: Fast non-dominated sorting process [244]

```
   Input: P = population of individuals
   Output: F_r = F1, F2, ... - set of non-dominated fronts
 1 F ← ∅
 2 foreach p in P do
 3     S_p = ∅
 4     n_p = 0
 5     foreach q in P do
 6         if p < q then
 7             S_P = S_P ∪ {q}
 8         else if q < p then
 9             n_P = n_P + 1
10     if n_p = 0 then
11         p_rank = 1
12         F_1 = F_1 ∪ {p}
13 i = 1
14 while F_i ≠ ∅ do
15     Q = ∅
16     foreach p in F_i do
17         foreach q in S_p do
18             n_q = n_q − 1
19             if n_q = 0 then
20                 q_rank = i + 1
21                 Q = Q ∪ {q}
22 i = i + 1
23 F_i = Q
24 return F
```

tive. The crowding distance for *i* is calculated as the sum of average side-lengths of the cuboid (dashed box) that is formed using its neighbouring individuals, *i-1* and *i+1*, as vertices [244]. The steps illustrated in Algorithm 2 are used to calculate the crowding distance of each individual in the Pareto front F, composed of N individuals. Each individual in F is initialised to 0 (line 2). The m in line 4 is used to index each objective. The $F[i]m$ is the *mth* objective of the *i*th individual in the set, F. f_m^{max} and f_m^{min} represents the maximum and minimum values for objective m [244]. For each objective m, the individuals in F are sorted in ascending order of magnitude based on their value for that objective (line 6). An infinite distance value is assigned



(a) Non-dominated sorting, where the population is divided into ranks

(b) Crowding distance calculation

Figure 4.5: Operations of NSGA-II algorithm [244]

to the individual with smallest and largest objective values (line 6). The distance of remaining individuals, that is *i=2 to (N-1)*, are calculated according to line 8, computing the normalised difference between the two nearest neighbours for the current objective. The total crowding distance is equal to the sum of the individual distance values of each objective [244].

Algorithm 2: Crowding distance procedure in NSGA-II [244]

Input: $N = |F|$
1 **for** *i=1...N* **do**
2 $\quad F[i]_{distance} = 0$
3 **for** *m=1...M* **do**
4 $\quad F = SORT(F,m)$
5 $\quad F[1]_{distance} = F[N]_{distance} = \infty$
6 $\quad$ **for** *i=2..(N-1)* **do**
7 $\quad\quad F[i]_{distance} = F[i]_{distance} + (F[i+1]m - F[i-1]m)/(f_m^{max} - f_m^{min})$

NSGA-II uses a crowded-comparison operator $<_c$, that guides the selection process during the evolution [244]. Each individual in the population is assigned a non-domination rank (i_{rank}), and a crowding distance ($i_{distance}$). The partial order $\prec_c$ is defined as follows [244]:

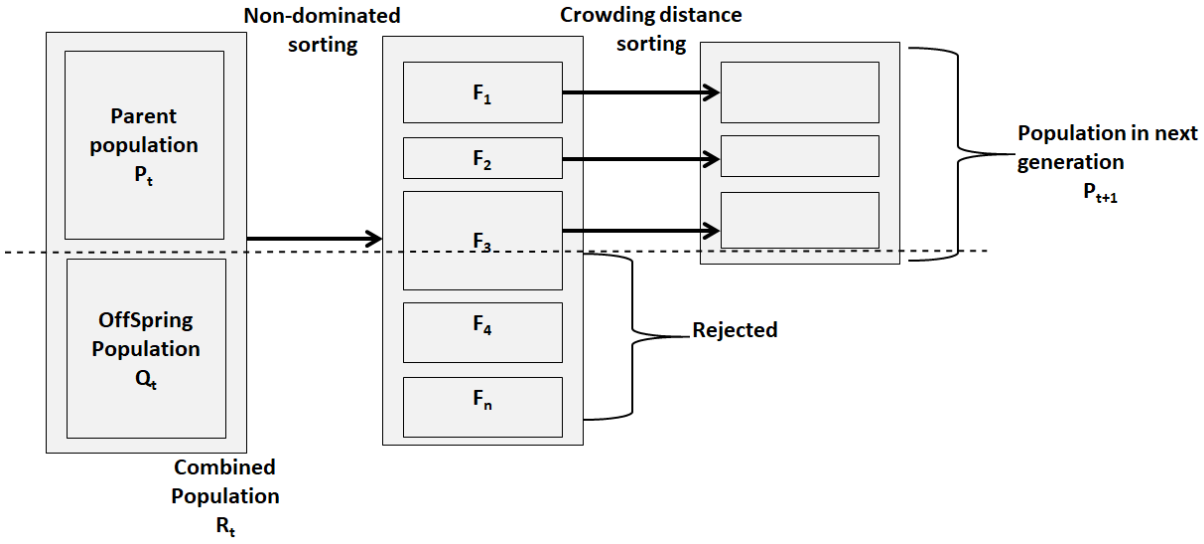


Figure 4.6: NSGA-II algorithm [244]

$$i \prec_c j := (i_{rank} \prec j_{rank}) \vee (i_{rank} = j_{rank} \wedge i_{distance} > j_{distance}) \tag{4.12}$$

That is, for two individuals with different non-dominations ranks, preference is given to individual with lower/better rank [244]. If both individuals are of the same rank, then the individual that is positioned in the less crowded region (has the higher crowding distance), is preferred [244]. This is to maximise the spreading of the individuals, thereby enforcing better diversity in the population.

Figure 4.6, which is adapted from [244], shows a single step of the NSGA-II algorithm. The initial population is generated randomly, and sorted on the basis of non-domination. Then the genetic operators, binary tournament selection, crossover and mutation operators are used to select the offspring, Q, a population of size N. For the next population, as illustrated in Figure 4.6, the parents P_t and Q_t, of size N, are combined to generate a population R_t of size $2N$, which is sorted into k non-dominated fronts $F_i, F_{i+1}, ..., F_k$. Individuals in the best non-dominated set are the fittest individuals in the combined population, and these appear in the top fronts such as F_1. If the size of these fronts is smaller than N, they are added to the new population. In this example, F_1 and F_2 are added to the new population,

as $|P_{t+1}| < |F_1| + |F_2| < N$. However, there is not enough space to take all the individuals in F_3 and F_4. F_4 is rejected, because it is of lower rank, and there is only limited space to take some members of F_3. To determine which members to select, F_3 is sorted according to crowding distance, and only those with the highest crowding distance are added to the new population P_{t+1} to make it up to size,N. After this, P_{t+1} is used to produce the next N offspring, Q_{t+1}, using crossover and mutation. In NSGA-II, selection for mating is done using a binary tournament based on the crowded-comparison operator ($<_c$) [244]. This process is repeated until the termination conditions are satisfied.

4.3.2.2 Strength Pareto Evolutionary Algorithm (SPEA2)

The Strength Pareto Evolutionary Algorithm 2 (SPEA2) [228] is an extension of the previous version of the Strength Pareto Evolutionary Algorithm (SPEA) [246] with significant improvements intented to obtain a population that has greater diversity and precision. The SPEA algorithm maintains an external archive that is used to accumulate non-dominated individuals. The algorithm starts with an initial population and an empty archive, the first step involves copying all non-dominated individuals in the population to the archive [228]. Once these individuals are added to the archive, any individuals that become dominated by these new entries are removed [228]. If the size of the archive is greater than the predefined size, further individuals are removed from the archive based on a clustering technique [228]. After this update operation, each member of the archive and population are given a fitness value computed using the number of individuals that the current individual dominates [228]. Then, using a binary tournament, the individuals for mating (crossover and mutation) are selected from the union of population and archive [228]. The resulting offspring population is used to replace the old population, and this processes is repeated until the termination conditions are met.SPEA2 makes some improvements on this process in several ways. It keeps the archive size constant and, if the non-dominated individuals are unable to fill the archive, the remaining space is filled by dominated individuals. SPEA2 has a fitness assignment strategy in which individuals are ranked on their raw fitness since the archive now contains domi-

nated individuals. Finally, another change introduced in SPEA2 is that mating takes places only amongst the members of the archive.

The steps involved in the SPEA2 algorithm are illustrated in Algorithm 3 [228]. The description and notation used in this section are taken from [228].

Algorithm 3: Procedure of SPEA2 [228]

Input: N = Size of Population, P
$\overline{N}$ = Archive Size
T = maximum number of generations
Output: ND - Set of non-dominated individuals

```
1  ND = ∅
2  P_0 = Generate initial Population randomly
3  t = 0
4  P̄_0 = ∅
5  while t < T do
6      Evaluate fitness of individuals in P_t and P̄_t
7      P̄_{t+1} ← non-dominated individuals in P_t and P̄_t
8      if |P̄_{t+1}| > N̄ then
9          Remove individuals in P̄_{t+1} using the Truncation Operator
10     else if |P̄_{t+1}| < N̄ then
11         Insert dominated individuals from P_t ∪ P̄_t
12     if t=T then
13         ND = P̄_{t+1}
14     else
15         Apply binary tournament selection over P̄_{t+1}
16         Apply the crossover and mutation over P̄_{t+1}
17         t = t + 1
18 return ND
```

The fitness assignment used by SPEA2 for each individual takes into account both the dominating and the dominated individuals. Every individual in the archive $\overline{P}_t$ and the population P_t has a strength value $S(i)$, denoting the total number of individuals it dominates. This is defined as [228]:

$$S(i) = |\{j | \in P_t \cup \overline{P}_t \wedge i \prec j\}| \tag{4.13}$$

where $i \prec j$ denotes the dominance relation. Based on the strength $S(i)$, the *raw fitness* $R(i)$ of an individual i is computed as follows [228]:

$$R(i) = \sum_{j \in P_t \cup \overline{P}_t, \wedge j \prec i} S(j) \tag{4.14}$$

This means that the raw fitness of i is calculated by summing the strength of its dominators in the archive and the population [228]. If $R(i) = 0$, then i is a non-dominated individual, and a high value of $R(i)$ corresponds to an individual i that is dominated by high number of individuals.

SPEA2 uses a density function to choose between individuals when they do not dominate each other. It computes the density using the classic *kth* nearest neighbour method [247], where the density of an individual is a function of the distance to the *kth* nearest individuals [228]. The distance of every individual i to all other individuals j in the population and archive is computed and saved in a list. Then, this list is sorted in ascending order and, for each individual i, the *k-th* nearest element yields the distance sought (σ_i^k). The recommended setting for k is the square root of combined sum of the population and archive size, $k = \sqrt{|N| + \overline{|N|}}$ [247]. Then, the density value for individual i is calculated as [228]:

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{4.15}$$

The value 2 is used in the denominator to ensure that $0 < D(i) < 1$. Finally, the total fitness value for each individual i is obtained by adding both raw fitness R and the density, D values [228]:

$$F(i) = R(i) + D(i) \tag{4.16}$$

A new archive is made up of all non-dominated individuals with fitness less than 1 from archive and population [228]:

$$\overline{P}_{t+1} = \{i | i \in P_t \cup \overline{P}_t \wedge F(i) \prec 1\} \tag{4.17}$$

If there is still space in the new archive, that is $|\overline{P}_{t+1}| < \overline{N}$, then the best dominated individuals in the previous population and archive can be copied into the new

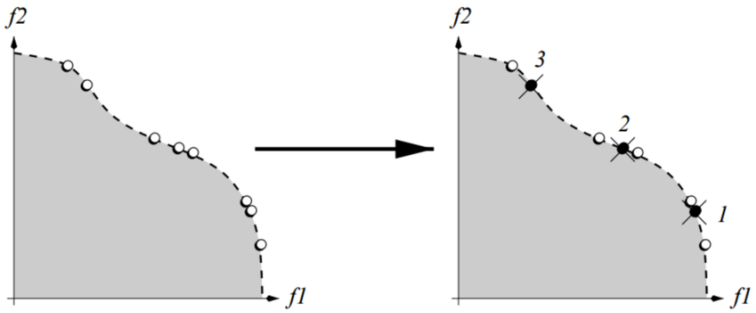


Figure 4.7: Example of the archive truncation method used in SPEA2 with $\overline{N}$, image taken from [228]

archive. If the size of the new archive is too large, ($\overline{P}_{t+1}| > \overline{N}$), individuals are truncated in overcrowded regions until $|\overline{P}_{t+1}| = \overline{N}$, based on their distance σ_i^k to other individuals. At each iteration of the truncation, the individual that has the smallest distance to another individual is removed [228]. Figure 4.7 shows an example of the truncation method. Assuming an archive size $\overline{N}$, the figure on the right illustrates the truncated individuals in the order of removal using the truncate operator.

4.4 Machine Learning Methods

Previous work found in the literature shows that there is a wide variety of techniques that have been applied to detect faults, and a few that have been used to detect attacks on the TE process model. Making a reliable comparison of these studies is not possible due to a lack of common metrics used to measure performance (sometimes none), different datasets and sizes. Furthermore, the nature of our data (attacks rather than faults) and the fact that the size of our data is much larger than the examples in the fault detection and diagnosis studies makes the problem more challenging. According to the *no-free lunch theorem*, there is no universal detection algorithm that is superior for all tasks, thus the performance of these algorithms is task-dependent. As a result, we decided to test a range of classical machine learning models, and some new techniques from deep learning, in attack detection.

The goal of machine learning is to learn patterns in the data, such as classifying the data into categories [248]. It draws from many fields including statistics, artificial intelligence, information theory, biology, psychology, cognitive science and

control theory [249]. Machine learning has been widely and successfully used in many applications of modern society: from web searches to autonomous vehicle training to cancer prognosis and prediction. In the information security field, machine learning has become a common tool for anomaly detection, spam filtering and malware classification. Since the beginning of 2000, there have been important advances in the theory and practice of the field, because of the creation of a class of techniques called deep learning [250].

Machine learning systems are classified based on the type and amount of data. There are four main classes: *supervised learning*, *unsupervised learning*, *semi-supervised learning* and *reinforcement learning*. In *supervised learning*, the learning uses training data that is composed of examples of the input vectors and their corresponding labels [248]. The tasks to which this is applied can be *classification*, assigning each input vector to a finite set of discrete categories [248], for example a spam filter with categories spam or not-spam. If the task is to predict a desired output as a continuous variable, for example price of a house, then the task is known as *regression* [248]. Often, especially in the security domain, the availability of labelled data is a problem or the process of labelling is expensive and prone to errors. In such cases the training data may not have any corresponding targets (labels), and the learning is done on unlabelled data, using a technique called *unsupervised learning*. The goal of this kind of learning includes [248]: grouping similar examples (clustering); determining the distribution of the data within the input space called density estimation; or dimensionality reduction (simplifying the data to a lower dimensional space by reducing the number of features without losing too much information). In cases, in which there is not large amount of labelled data, but some small amount of labelled data is available, then *semi-supervised learning* models can be used to train the models using the labelled data,incorporating with unlabelled data using the labels predicted from the trained model. Finally, *reinforcement learning* has a different learning goal, is to take actions that maximise rewards [251]. Here, the learning system is called an *agent*, which observes the environment, selects and carries out actions and gets a reward or penalties (negative

rewards). Thus, the algorithm is trying to learn a policy that defines actions that are taken when the system is in a particular state. Some popular applications of reinforcement learning include learning to play board games, *tabula rasa*, to a very advanced level [252], optimising manufacturing [253] and robotics [254].

Supervised Learning	Unsupervised Learning
Single Decision Tree (CART)	One-Class SVM
Ensemble Learning (Bagging, Boosting, Random Forest) using CART	Recurrent Neural Network (LSTM, GRU)
Support Vector Machine	Autoencoder Neural Network

Table 4.5: Selected supervised and unsupervised models for detection

To investigate the effectiveness of supervised learning (classification) and unsupervised learning (anomaly detection) we selected some of the approaches that have been used in the past. Supervised learning requires both labelled data containing attacks and normal operational data, and it suffers from several limitations: obtaining attack data is often difficult; the process of labelling the data may be prone to errors or may be expensive; and performance deteriorates significantly for unknown attacks [255]. Given the limitations of supervised learning, there is an increasing move towards generating unsupervised learning approaches for ICS security, but the effectiveness of these approaches in practice requires further study. Unsupervised learning models work on unlabelled normal operating data and from, this learn what is normal; detection then involves deciding what is not normal. Table 4.5 shows the learning models selected for further investigation. SVM and One-Class SVM were selected because they were widely used in the literature for detecting both faults and attacks. However, they tend to be slow and computationally expensive on large datasets. Therefore, they may not be suitable for real ICS systems with large datasets. Decision tree models often perform well on large datasets and, therefore, one of the most popular decision tree algorithms, Classification And Regression Trees (CART) [256] algorithm was selected for further investigation. The ensemble method is often used with decision tree algorithms (combining multiple decision trees to improve the performance of the classifier over a single tree).

Taking advantage of large datasets and efficient algorithms, deep learning tech-

niques have been effective in solving problems in many disciplines including computer vision, speech and voice recognition, natural language processing, robotics, bioinformatics, healthcare, finance, and advertising [257] using supervised and unsupervised learning. They are currently attracting an increasing amount of research to investigate their application to solution of security tasks. However, further work is needed to compare them to more conventional methods to understand the benefits of these approaches. Two powerful forms of Recurrent Neural Network (RNN), LSTM and GRU, with the addition of an autoencoder neural network were selected for anomaly detection as a comparator for One-Class SVM. LSTM and GRU have been applied to predict unsupervised anomaly detection in time series. Autoencoders are unsupervised neural metwork models that have been used to learn meaningful features in datasets. They can outperform conventional methods like PCA on tasks like reducing the dimensionality of complicated datasets [258]. Some of their applications include classification, clustering and generative problems. They can be useful for modelling important features of complex and larger datasets in ICS. Details of all these models will be discussed in details in the following subsections.

4.4.1 Decision Trees

Decision trees are one of the most powerful and widely used supervised learning techniques and are used for classification and regression tasks in medicine, science and engineering. They are easy to understand and implement, are computationally cheap to build, and do not require special data preparation. These models work by recursively partitioning the input data space into subset regions, and then assigning a prediction model to each region [248]. These partitions correspond to a sequential decision making process, which is represented as a decision tree with a set of *if-then-else* decision rules, starting from the top of the tree and travelling to the leaf nodes.

The Classification and Regression Tree (CART) [256] algorithm is one of the most widely used algorithms in training decision trees, and it is the algorithm that is used for decision trees classification in this thesis. A Python implementation is available in scikit-learn [92] and was used in this work. However, there are other

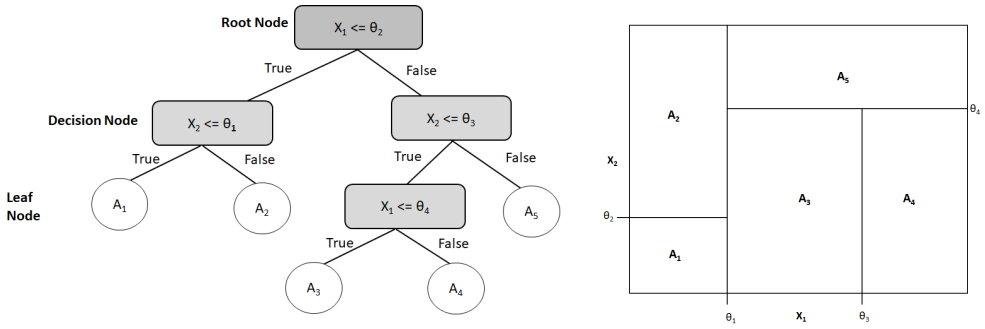
variants of decision tree algorithms including ID3 [259] and C4.5 [260].

The CART algorithm uses a greedy top-down binary recursive partitioning technique that divides the data space into heterogeneous cuboid regions [256, 248]. The term binary implies that each data set can be split into two subsets. Figure 4.8b, adapted from [248], provides an instance of the recursive partitioning of the input data and the corresponding decision tree structure with two X variables and five classes (A-E). It splits the input data into two regions based on $(X_1 \leq \theta_1)$ or $(X_1 > \theta_1)$, where θ_1 define the parameter of the model [248]. These two subregions can then be further divided. For example, the region corresponding to $(X_1 > \theta_1)$ is further divided into two other regions based on $(X_2 \leq \theta_3)$ or $(X_2 > \theta_3)$, giving rise to a region E, and a region that is further subdivided to give rise to the regions denoted C and D [248]. Since the decision trees will be used for classification, each of the regions will correspond to a specific class [248]. This process then can be described as a binary decision tree, as illustrated in Figure 4.8a. The decision tree starts from the *root node* which takes in the entire input data or sample and splits it into two *child nodes*, creating the first decision condition $(X_1 <= \theta_1)$ to do so [248]. Child nodes can be further divided into further child nodes, known as *decision nodes* [248]. Nodes that do not split into further decision nodes are called *leaf* or *termination* nodes, representing the predicted class value [248]. The *depth* of the tree is defined as the maximum length of a path from the root node to a leaf node [248]. Each path from the root to a leaf node can be transformed into a rule by joining the parameters and conditions tested at each child along the path [248].

To build a decision tree model from the training set, the CART algorithm needs to determine how to split the node into two child nodes. Starting from the input training data (root node), CART considers all the input variables (features), and selects the best split on a cost function that aims to decrease the *impurity* to the greatest extent possible. The decrease in the impurity is defined as [256]:

$$\triangle i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R) \tag{4.18}$$

where $\triangle i(s,t)$ is the impurity at parent node t with split s, $i(t)$ is the impurity



(a) Decision Tree **(b)** Decision Boundaries

Figure 4.8: An example of a partitioned two dimensional input space and the corresponding decision tree [248]

of t, p_L and p_R are probabilities of data instances for the left child t_L and right child t_R nodes and $i(t_L)$ and $i(t_L)$ are the impurities of the left and right child node[256]. There are various purity measures that are used by decision trees including Gini impurity, entropy impurity, Chi-square test, variance reduction and misclassification measure. The implementation of the CART algorithm used in this thesis comes with two possible measures: Gini impurity and entropy impurity. However, the trees obtained are usually the same. The default Gini impurity measure was selected for the work carried out in this thesis because it is slightly faster than using entropy. The Gini impurity measure of the node t is defined as [256]:

$$I(t) = 1 - \sum_{k=1}^{K} {p_{t,k}}^2 \tag{4.19}$$

where $p_{t,k}$ is the ratio of class k instance among the training instances of node t. When the Gini impurity measure is zero, all instances in the node fall into the same class, and the node is *pure*.

This splitting process continues recursively until it reaches some stopping criterion. Two of the common stopping conditions are *maximum tree depth* and the point at which it fails to find a split that will reduce the impurity. As for all other machine learning models, overfitting is one of the weaknesses of decision trees. In this case, the tree structure adapts itself to fit all samples in the training data, losing its generalisation capabilities to new data. Decision trees are defined as *non-parametric* models. This means that the model makes no assumptions about the

distribution of the data, which may cause the model to align itself too closely with the training data. To reduce the risk of overfitting the training data, it is recommended that constraints are placed on the training process. Generally, this is done by restricting the maximum depth of the tree. However, the growth of decisions tree can be further controlled using other parameters, some of which are defined also by the Scikit-Learn package [92], such as the minimum number of instances required to be a leaf node; maximum number of leaf nodes in a tree; and minimum number of samples required to split a node to improve the performance of the decision trees.

4.4.2 Ensemble Learning using Decision Trees

Ensemble learning is a powerful machine learning technique in which a group of learning algorithms (*predictors*) are used to solve the same problem and, when combined, produce better predictions than using a single learning algorithms. Using a group of multiple predictors is known as an *ensemble* [261]. There are several techniques used to achieve this.

A simple approach is to use the same training data with different classifiers such as decision tree, SVM and Logistic Regression. One then outputs the result with the most votes. A second approach uses a single classification algorithm for every predictor and trains them using random subsets of the training data. In this thesis, the second approach has been leveraged to test the performance of ensemble learning using the CART algorithm. The techniques used are *Bagging*, *Random Forest* and *Boosting* (AdaBoost algorithm) techniques.

In *Bagging* [262], multiple predictors (in our case decision trees) are used train a group of classifiers to reduce the variance of the predictions. It is a technique widely used with high-variance and low bias learning models, and this includes decision trees [263]. Bagging uses random sampling to generate a subset of data for each classifier. Once all predictors are trained, the instances of new data can be tested by asking all individual predictors for a prediction, and aggregating these to obtain an overall prediction. This aggregation involves taking the mode of the individual predictions. Bagging is a general technique that can be used with learning methods other than decision trees.

Random Forest is an ensemble of decision trees using bagging method with some improvements to ensure the growth of diverse trees. Decision trees consider all features in order to split a node and select the best; however, Random Forest searches for the best feature from a subset of features selected at random [264]. This increases the diversity of the tree and, in general, improves the performance of the classifier and provides better control on overfitting. Experiments were carried using both Random Forest and Bagging with CART to determine the performance difference.

Boosting [265] is another powerful ensemble learning technique used in machine learning and is suitable for classification tasks. The motivation for the boosting technique was to combine multiple weak learners to produce a strong one. Boosting works by training predictors iteratively, each current predictor trying to improve on its predecessor [264]. The most popular and widely used Boosting algorithm is *AdaBoost* (Adaptive Boosting) [265]. It is also the algorithm used in this thesis with decision trees. AdaBoost is often used with weak classifiers (e.g. one whose accuracy is slightly better than random guessing); however they can be applied to any classifiers. Our investigation aimed to determine if there were any benefits of using boosting with decision trees.

Algorithm 4: Procedure of Adaptive Boosting [265]

Input: Given the training data = $(x_1, y_1), ...(x_m, y_m)$ where
$x_i \in X, y_i \in \{-1, +1\}$

Initialise the set of weights $w_1(i) = 1/N$

for $t = 1$ **to** T **do**

  Train a base classifier $G_t(x)$ using weights $w(i)$

  Get weighted error rate $\varepsilon_t = \frac{\sum_{i=1}^{N} w_i I[y_i \neq G_t(x_i])}{\sum_{i=1}^{N} w_i}$

  Compute $\alpha_t = log\frac{1-\varepsilon_t}{\varepsilon_t}$

  For $i = 1, 2, ..N$ update weights: $w_i \leftarrow w_i.exp(\alpha_t.I(y_i \neq G_t(x_i))]$

Output the final classifier $G(x) = sign(\sum_{t=1}^{T} \alpha_t G_t(x))$

The AdaBoost algorithm starts with a base classifier (in our case the decision tree) and iteratively forms a number of classifiers [264]. It maintains the relative weights of misclassified training instances. Initially these weights are equal but,

at successive iterations, the weight of the missclassified instances are increased so that the next classifier will pay more attention to the underfitted instances that were missed in the previous iteration. Algorithm 4 shows the steps for the AdaBoost. Given a training dataset X with N instances, consider a classification problem with two classes, $Y \in -1, 1$. Initially, each instance of X is set to $1/N$ and the classifier trains on the data in the usual manner. Then the current classifier $G_t(x)$ is run on the weighted instances, and the resulting weighted error rate for $G_t(x)$ is computed (line 3). Line 4 computes the weight $\alpha(t)$ assigned to $G_t(x)$ in producing the final classifier (line 7). Next, the instance weights are updated ready for the next iteration, and the missclassified instances are scaled by a factor $exp(\alpha_t)$, boosting their influence on the next classifier $G_{t+1}(x)$. The algorithm terminates when the desired number of classifiers (T) is reached, and the algorithm outputs the final classifier. To make new predictions, the algorithm gathers the predictions of all the classifiers and weights their contribution using classifier weights $\alpha(t)$. The final class prediction is based on a weighted majority vote: that is, the class that has the highest total weight is the predicted class.

The most important parameter to adjust when using ensemble algorithms is the *number of estimators*; in our case, this is the number of decision trees in the ensemble.

4.4.3 Support Vector Machines

Support Vector Machines (SVMs)[266] [267] are one of the most powerful approaches to supervised learning and are widely used in for classification, regression and outlier detection tasks in many real-world applications, including security. In this section, we introduce how SVM is used to solve classifications problems, and later we will introduce how SVM can be used for outlier detection.

SVMs are inherently two-group (binary) classifiers, and they have been used to model a binary classifier in this thesis. However, it is worth noting they have been also extended to multi-class classifications using a variety of approaches [268]. The description and notation used in this section are taken from [269, 270]. SVM aims to solve classification problems by determining a good decision boundary between two

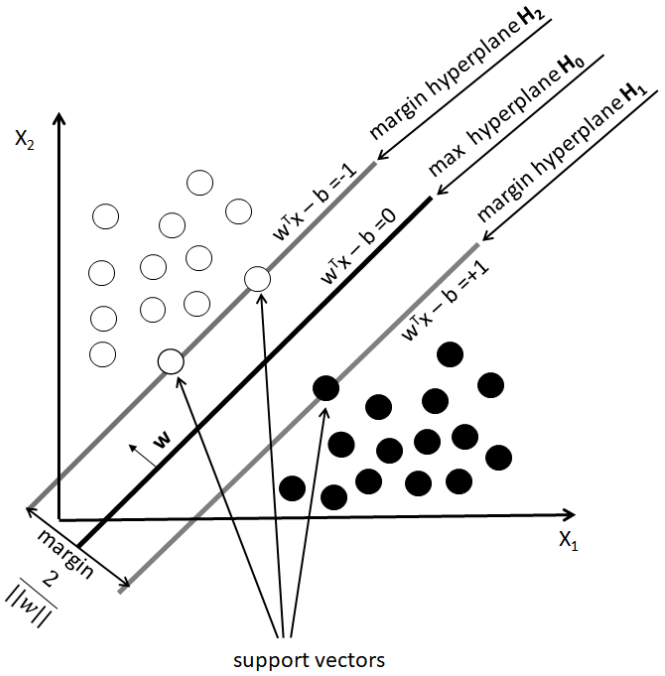


Figure 4.9: A binary SVM classifier with maximal margin hyperplane (adapted from [271])

sets of data belonging to two different classes. Figure 4.9 describes the fundamental idea behind the SVM classifier using a linearly separable binary set. The aim of the SVM is to determine the best hyperplane that can be used to separate the instances of the training data into two classes. This involves finding the widest margin between the two classes. In the SVM literature, these two classes are commonly denoted as $+1$ and -1. The SVM is given x as the training samples, where each instance in the training set has n dimensions, and a class label with two possible values $y \in \{-1, 1\}$. The separating hyperplane is defined by the equation $\mathbf{w^T x} + b = 0$, and relies on a weight vector $\mathbf{w}$ and a bias parameter b that separates the inputs of the two classes.

Based on this, the ith sample $\mathbf{x_i}$ is predicted to belong to class positive if $\mathbf{w^T x_i} + b$ is positive, and to class negative if $\mathbf{w^T x_i} + b$ is negative. Given the maximal margin hyperplane $(\mathbf{w}, \mathbf{b})$ that separates the training data, the decision function [270]:

$$f(\mathbf{x}) = sign(\mathbf{w^T x} + b) \tag{4.20}$$

assigns the training data to their classes.

Figure 4.9 [271] shows the decision boundary in SVM, which is a separating

hyperplane represented as the dark line in the middle (H_0). The elements above this hyperplane belong to class -1, and the elements below the hyperplane belong to class $+1$. The goal of the SVM is to select the hyperplane that gives the maximal geometric distance between the hyperplane and the closest points from either class (known as support vectors) as shown by the two grey lines (H_1 and H_2) which represent the $\mathbf{w^T x_i} + b = -1$ and $\mathbf{w^T x_i} + b = 1$. As shown in Figure 4.9, this distance is the total distance between H_1 and H_2. Recall that in vector algebra, the distance from the origin along the direction $\mathbf{w}$ to a point $\mathbf{x}$ is defined as [270]:

$$\frac{\mathbf{w^T x}}{\sqrt{\mathbf{w^T w}}} \tag{4.21}$$

Based on this, which is the *margin*, the distance between H_1 and H_2, is calculated as [270]:

$$\frac{\mathbf{w^T}}{\sqrt{\mathbf{w^T w}}}(x_+ - x_-) = \frac{2}{\sqrt{\mathbf{w^T w}}} = \frac{2}{\|\mathbf{w}\|} \tag{4.22}$$

Maximising the distance between the two margin hyperplanes is done by minimising the length $\mathbf{w^T w}$. This is transformed into an optimisation problem such that [270]:

$$\textit{minimise}_{w,b} \quad \frac{1}{2}\mathbf{w^T w} \quad \text{subject to } y^i(\mathbf{w^T x^i} + b) \geq 1, \quad \forall i \in \{1, \ldots, N\} \tag{4.23}$$

In addition, each instance $\mathbf{x^i}$ in the training set is assigned a label $y^i \in -1, +1$ and the distance between the two hyperplanes (H_1 and H_2) is maximal. This makes it a *quadratic programming* problem. The classifier defined so far, known as the *maximal margin classifier* or *hard-margin* SVM [269], makes the assumption that the data is linearly separable and there are no training errors; however, this is often not the case in the real-world, where many problems contain noise and outliers. In these cases, finding a hyperplane that will completely separate all training data into the correct classes may not be possible, and one might prefer a more robust solution that separates the majority of the data while ignoring some outliers. To account for

these cases, *slack variables*, ξ^i, are used to relax the margin constraints [270]:

$$y^i(\mathbf{w}^\mathbf{T}\mathbf{x}^\mathbf{i}+b) \geq 1-\xi^i, \quad \forall i \in \{1,\ldots,N\}, \qquad \xi^i \geq 0, \quad \forall i \in \{1,\ldots,N\} \tag{4.24}$$

As illustrated in Figure 4.10, if the sample point ξ^i has a value $0 < \xi^i < 1$ then it is on the correct side of the decision boundary; however, if $\xi^i > 1$ then the data instance is of the opposite class [270]. Each x^i has its own slack variable, and this is considered when solving the optimisation problem such that [270]:

$$\begin{aligned} \textit{minimise}_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^\mathbf{T}\mathbf{w} + C\sum_{i=1}^{N}\xi^{(i)} \\ \text{subject to} \quad & y^{(i)}(\mathbf{w}^\mathbf{T}\mathbf{x}^{(i)}+b) \geq 1-\xi^{(i)}, \qquad \forall i \in \{1,\ldots,N\} \\ & \xi^{(i)} \geq 0, \qquad \forall i \in \{1,\ldots,N\}. \end{aligned} \tag{4.25}$$

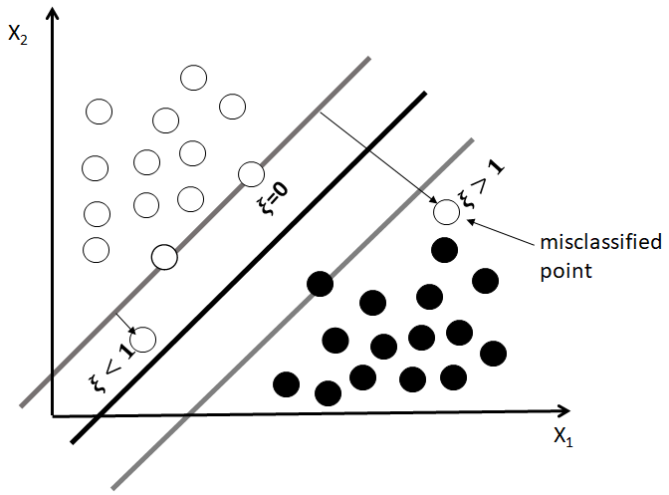


Figure 4.10: Slack variable for the SVM (adapted from [248])

In this, C is the parameter that allows some data points to be misclassified. C is a tunable parameter that is usually selected based on testing a wide range of values by using search techniques and by validating on separate test data. A higher C corresponds to greater importance in classifying all the training samples correctly and a lower C can make the optimiser to search for a larger-margin separating hy-

perplane, which may result in misclassifying training data samples. This concept is known as *soft-margin* optimisation, and there are two standard approaches to it, namely *2-Norm Soft Margin* and *1-Norm Soft Margin* [269] using duals.

4.4.3.1 Soft Margin Optimisation

The optimisation problem introduced in the previous section is known as the *primal problem*. This problem can be reformulated into a related problem called its *dual problem* that builds a lower bound of the primal problem while ensuring that the optimal solution is the same as the primal problem. Transforming the primal problem to the dual problem allows classification problems to be solved faster especially in situations where the dimensionality of the data (that is the number of features) is larger than the number of data instances. Furthermore, dual representation is required for the *kernel trick* technique which will be discussed shortly. The *Lagrange Multipliers* method is use to derive the dual problem from the primal problem. So as to not lose focus in this chapter, we refer the readers interested in the full mathematical details of this transformation to [269]. Here we only the show the transformation for 1-norm soft margin technique as it tends to be used more often in practice. The optimisation problem *primal* in Equation 4.26 is represented using the Lagrangian [269]:

$$L(\mathbf{w},b,\xi,\alpha) = \frac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} + C\sum_{i=1}^{N}\xi^i - \sum_{i=1}^{N}\alpha^i[y^i(\mathbf{w}^{\mathbf{T}}\mathbf{x}^i + b) - 1 + \xi^i] - \sum_{i=1}^{N} r^i\xi^i$$
$$\text{subject to } \alpha^i \geq 0, \xi^i \geq 0, r^i \geq 0 \qquad \forall i \in \{1,\ldots,N\} \tag{4.26}$$

where $\alpha^i \geq 0$ and $r^i \geq 0$ are multipliers.

The corresponding *dual* for the optimisation problem is found by differentiating the Lagrangian with respect to primal variables, $\mathbf{w}, \xi$ and b, and setting the results equal to zero as follows [269]:

$$\frac{\partial L(\mathbf{w},b,\xi,\alpha,r)}{\partial w} = \mathbf{w} - \sum_{i}^{N} y^i\alpha^i\mathbf{x}^i = 0 \tag{4.27}$$

$$\frac{\partial L(\mathbf{w},b,\xi,\alpha,r)}{\partial \xi^i} = C - \alpha^i - r^i = 0 \tag{4.28}$$

$$\frac{\partial L(\mathbf{w},b,\xi,\alpha,r)}{\partial b} = \sum_{i,j}^{N} y^i \alpha^i = 0 \tag{4.29}$$

From this, $\mathbf{w}$ is obtained by:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha^i y^i \mathbf{x}^i \tag{4.30}$$

Support vectors have non-zero values, $\alpha^i > 0$. Using these conditions and resubstituting Equations 4.27 to 4.29 back into the primal problem, the following dual problem is derived [269]:

$$L(\mathbf{w},b,\xi,\alpha,r) = \frac{1}{2}\alpha^i - \frac{1}{2}\sum_{i,j=1}^{N} y^i y^j \alpha^i \alpha^j (x^i)^T x^j \tag{4.31}$$

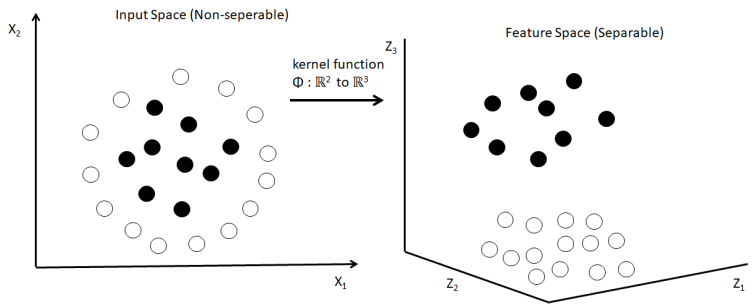


Figure 4.11: Mapping of inseparable training data from $\mathbb{R}^2$ to $\mathbb{R}^3$

To find the hyperplane of complex non-linear data sets, the input samples x are transformed to a higher dimensional feature space by applying a mapping function ϕ. When the data is mapped to a higher dimensional feature space, finding a separating linear hyperplane is easier as then data points are linearly separated, as illustrated in Figure 4.11. This is achieved by replacing x with ϕx, and can be written as follows [270]:

$$L(\mathbf{w},b,\xi,\alpha,r) = \frac{1}{2}\alpha^i - \frac{1}{2}\sum_{i,j=1}^{N} y^i y^j \alpha^i \alpha^j \phi(x^i)^T \phi(x^j) \tag{4.32}$$

The values for ϕ can be computed using a kernel, K, defined as [270]:

$$K(\mathbf{x}^i, \mathbf{x}^j) = \phi(\mathbf{x}^i)^T \phi \mathbf{x}^j \tag{4.33}$$

Adding the kernel function, the optimisation problem is simplified into [270]:

$$\begin{aligned} &\textit{maximise} \quad \sum_{i=1}^{N} \alpha^i - \frac{1}{2} \sum_{i,j}^{N} y^i y^j \alpha^i \alpha^j K(\mathbf{x}^i, \mathbf{x}^j) \\ &\text{subject to } \sum_{i}^{N} y^i \alpha^i = 0, \quad 0 \leq \alpha^i \leq C \qquad \forall i \in \{1, \ldots, N\} \end{aligned} \tag{4.34}$$

The optimisation problem learns α^i using the (x^i, y^i) of the training set. Once the optimal solution for α^i is learned, the decision function in Equation 4.20 becomes [270]:

$$f(\mathbf{x}) = sign(\sum_{i=1}^{N} \alpha^i y^i K(\mathbf{x}^i, \mathbf{x}) + b) \tag{4.35}$$

4.4.3.2 Kernel Trick

As mentioned previously, if the data is not linearly separable in the original space, SVM uses the *kernel trick* [272], which involves defining a mapping function that transforms the data from a lower dimensional to a higher dimensional feature space. It is worth noting that the input used in the dual Lagrangian so far uses scalar products $(\mathbf{x}^i)^T \mathbf{x}^j$. x can be mapped to a vector function of $\mathbf{x}$ [270]:

$$K(\mathbf{x}^i, \mathbf{x}^j) = \phi(\mathbf{x}^i)^T \phi(\mathbf{x}^j) \tag{4.36}$$

So, the *kernel* is the function that maps $x^i, \mathbf{x}^j$ to the inner product $\phi(\mathbf{x}^i)^T \phi(\mathbf{x}^j)$. Kernels are used to transform the input data into the required feature space, and then a linear classification is achieved in this new space. This technique is known as the kernel trick, and it makes it possible for the SVM to be generalised to nonlinear cases. Some of the commonly used kernel functions are [264]:

- Linear Kernel: $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$

- Polynomial Kernel: $K(\mathbf{x},\mathbf{y}) = (\gamma x^T \cdot \mathbf{y} + r)^d, \gamma > 0$
- Radial basis function (RBF) kernel: $K(\mathbf{x},\mathbf{y}) = \exp(\|\mathbf{x}-\mathbf{y}\|^2), \gamma > 0$
- Sigmoid kernel: $K(\mathbf{x},\mathbf{y}) = \tanh(\gamma \cdot x^T y + r)$

where r, d and γ are kernel parameters. The selection of the kernel depends on the problem, and the general recommendation is to experiment with different kernels (and kernel parameters) to search for the best for the problem at hand. However, using the *RBF* kernel (sometimes known as the Gaussian Kernel) and polynomaial kernels has been shown to be a good initial approach. The RBF kernel was selected for the SVM models in this thesis. This is based on performance of initial experiments and previous work carried out by others on the TE process model, which shows that if the right parameter for γ is selected, the RBF kernel performs well.

4.4.3.3 One-Class SVM

The original Support Vector Machine was been extended for unsupervised learning by Schölkopf et al [272]. The objective of this algorithm is novelty detection or outlier detection. Given a single class training dataset, the one-class SVM learns the decision boundary of this single class. Once it captures this decision region, new test samples can be used to determine whether they belong to this class. Data samples falling outside this boundary are known as *outliers*. In the context of intrusion detection, normal operational data is used to train the model to learn the boundary of the normal data. During the test phase, any samples that fall within the normal boundary are considered normal, otherwise they are classified as outliers or as anomalies.

Figure 4.12 illustrates the basic idea behind the One-Class SVM. After mapping the input data into feature space using some kernel, it is separated from the origin with maximum margin to determine the hyperplane. Once the optimal hyperplane is found, the class of the new data instance x can be determined. If x falls below the hyperlane, it is classified as class -1 (outlier), if it falls on or above the hyperplane it is belongs to class $+1$ (normal). The notation used here are adapted from the original description of the One-Class SVM [272].

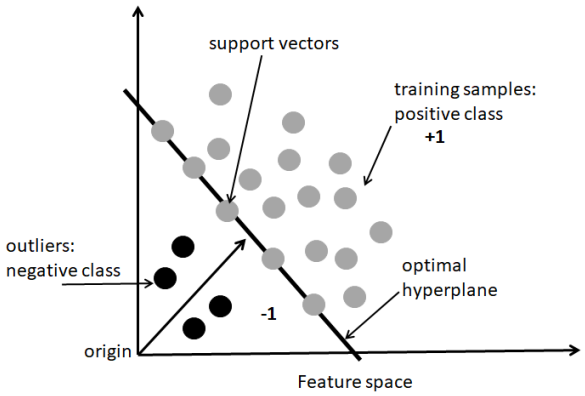


Figure 4.12: Data classification based on One-Class SVM

To separate the data, the One-Class SVM solves the following optimisation problem [272]:

$$\begin{aligned} &\textit{minimise}_{w,\xi^i,\rho} \quad \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi^i - \rho && (4.37)\\ &\text{subject to } (w\cdot\phi(\mathbf{x}^i)) \geq \rho - \xi^i, && \forall i \in \{1,\ldots,n\}\\ &\qquad\qquad \xi^i \geq 0 && \forall i \in \{1,\ldots,n\} \end{aligned}$$

where (w,ρ) are the weight vector and the offset, ξ^i is the slack variable for data i, n is the size of the training dataset, and ν is a regularisation parameter. In the binary SVM, the penalty parameter C was used to control the number of errors; in this equation, it is the parameter ν that sets an upper bound on the fraction of outliers in the training data and a lower bound on the number of support vectors. As in the binary SVM, this problem is converted into the dual problem using Lagrangian multipliers. For mathematical details please see [272], however as before after applying multipliers and selecting an appropriate kernel function ($K(x,y) = \phi(x)\cdot\phi(y)$), the decision function takes the following form:

$$f(x) = \text{sgn}(\sum_{i=1}^{n}\alpha_i K(x,x_i) - \rho) \tag{4.38}$$

That is, a data instance x belongs to class 1 if $f(x) \geq 0$ and class -1 otherwise.

4.4.4 Deep Learning and Deep Neural Network

In 2006, Hinton proposed a fast learning algorithm for deep structured networks called the deep belief network (DBF) [273]. Since then, interest in deep learning has undergone rapid growth. Deep learning is been successfully applied to problems in which traditional machine learning models have failed to produce satisfactory performance, such as image and speech recognition, computer vision, and natural language processing; in all of these, the training data is noisy and complex. Deep learning builds on the concept of Artificial Neural Networks (ANNs) composed of multiple layers. The work, 'A Logical Calculus of the Ideas Immanent in Nervous Activity' [274], by neurophysiologist Warren McCulloch and logician Walter Pitts in 1943 is considered to be the starting point for ANNs. McCulloch and Pitts presented a model of how biological neurons might work together to perform computational functions. ANNs have received significant research attention over the intervening years; however, they have become substantially more popular over the last decade or so. This is mainly due to availability of data (they require large datasets to work well); an increase in available computing power to train them in reasonable time; and an increase in research funding from the public and private sectors to carry out more practical research.

The mammalian brain contains over 100 billion interconnected neurons, each of these neurons consists of a cell body, dendrites, and an axon [275]. The neuron activity is inhibited through connections to other neurons. The axons are like wires that extend from the cell body, and split off into branches before ending into many axon terminals. These terminals at the end of axons connect to other neurons to transmit electrical signals. Dendrites sprout from the neuron cell body and serve as receptors for receiving incoming signals from other neurons, and relay them to the cell body for a response. When a neuron receives a strong signal, it fires its own signals. Each individual neuron has the ability to form thousands of synaptic connections and, overall, to form a vast network of billions of neurons, which effectively gives humans the ability to process information in the way they do. ANNs are a simplified representation of this behaviour.

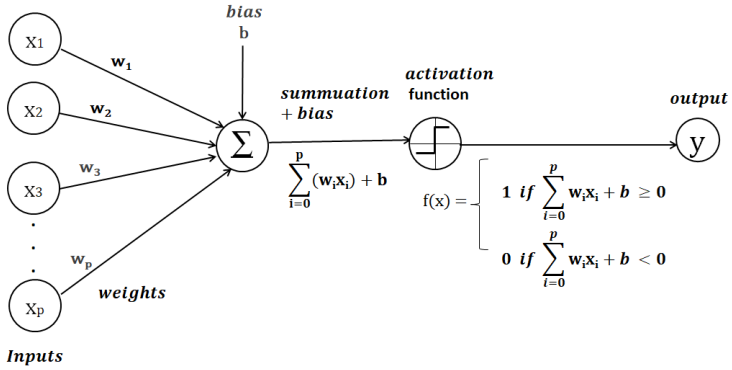


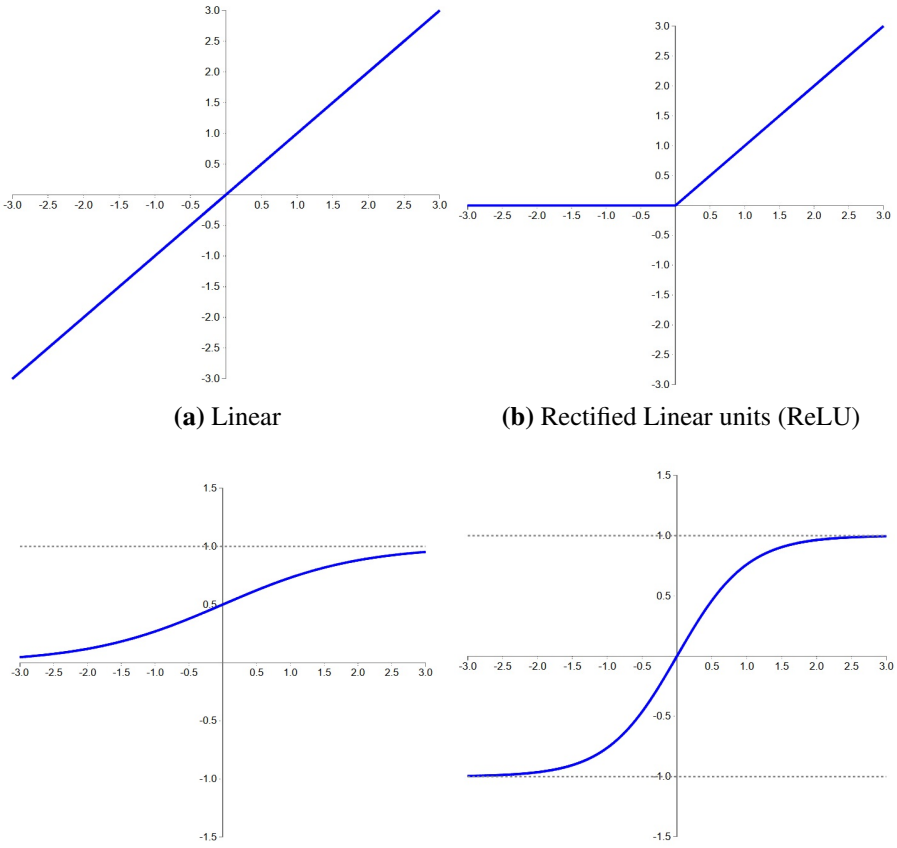
Figure 4.13: A single-layer perceptron network (adapted from [264])

The simplest type of ANN is called a *perceptron*. As shown in Figure 4.13 the behaviour of the perceptron is similar to the biological neuron [264]. It takes several inputs, multiplies them by a set of weights and a bias, and then sums all this together and feeds into an activation function. There are a number of common activation functions that are used with ANNs. Figure 4.13 shows a unit step function, that output a 1 if the calculated value is greater than a threshold, and 0 otherwise. This simple functionality can be effectively used for binary classification, in which one wants to classify inputs into two groups. More specifically, each input, x_i to x_p is multiplied by a weight w_i, and all the values are added:

$$f(x) = \begin{cases} 1 & \text{if } w_1x_1 + w_2x_2 + ... + w_px_p + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.39}$$

The weights determine the contribution of the input x_i to the perceptron output y. Often, an additional constant value, bias, b, is added to the weighted sum to enhance the activation. The result is used by the activation function to determine the output of the neuron.

Learning in a perceptron requires one to determine the values for the weights w_i to w_p and the bias, b, all of which are done by training. There are a number of activation functions in use with artificial neural networks. Some of the most common are logistic, sigmoid, and tanh (or hyperbolic tan), and their behaviours are illustrated in Figure 4.14:



(a) Linear **(b)** Rectified Linear units (ReLU)

(c) Sigmoid **(d)** Tanh (or Hyperbolic Tan)

Figure 4.14: Common ANN activation functions

$$\text{linear: } f(x) = x \tag{4.40}$$

$$\text{ReLu: } f(x) = \begin{cases} x & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases} \tag{4.41}$$

$$\text{sigmoid: } f(x) = \frac{1}{1+e^{-x}} \tag{4.42}$$

$$\text{tanH: } f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1 \tag{4.43}$$

Stacking multiple layers of perceptrons is known as *multilayer perceptrons* or *feedforward neural network* (FNN) [276]. These networks are composed of an *input*

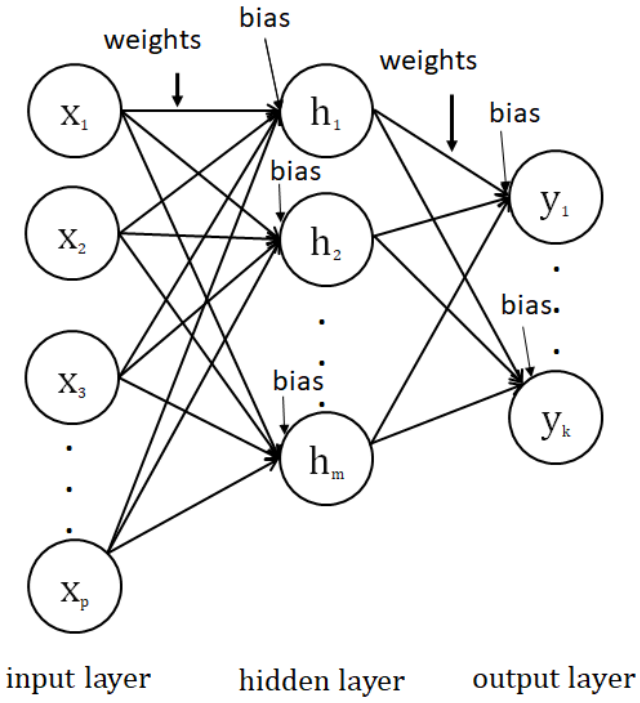


Figure 4.15: A three-layer feedforward neural network [276]

layer that receives the input, one or more *hidden layers* and an *output layer* that produces the output of the neural network. An example of a three layer feedforward neural network is illustrated in Figure 4.15. Computing the output for neuron h_1 in the hidden layer h^j, (in this case there is one hidden layer, $j = 1$) involves computing the dot product for input data $X : x_1, x_2 ... x_p$, with the weight of the neurons of the layer, $W^l = w^l{}_1, w^l{}_2, .. w^l{}_p$ and adding the bias. Computed as:

$$z = \sum_{i=1}^{p} w_i x_i + bias \tag{4.44}$$

By feeding z into the activation function, $\phi(z)$, the output for the hidden layer's first neuron is obtained. This procedure is repeated for all the other neurons in the hidden layer, from h_2 to h_m. These values are then used as inputs to calculate the values of the output layer:

$$y_i = \phi(\sum_{i=1}^{m} w_i^{h^j} h^j{}_i + bias) \tag{4.45}$$

4.4.4.1 Training an Artificial Neural Network

Training an ANN is an optimisation problem with the intention of minimising the loss function, which measures the inconsistency between actual and predicted outputs. This is achieved by tuning the parameters of the network, weights and biases (θ) so that the loss is minimised. The most common loss function used for regression task is mean squared error:

$$J(\phi) = \frac{1}{N}\sum_{i=1}^{N}(y_{(i)} - \hat{y}_{(i)})^2 \tag{4.46}$$

where N is the number of observations, y_i is the actual value, and $\hat{y}_i$ is the predicted value. ANNs use a *gradient descent* algorithm [277] to calculate the gradient of the loss function $J(\theta)$ with respect to the θ. In ANNs, gradients are computed using *back propagation* [277]: that is, the gradient calculation operation begins from the final layer and proceeds backwards to the first layer of weights to make the appropriate weight and bias changes in such a way that the error is minimised. Figure 4.16 illustrates the operation of the gradient descent algorithm.

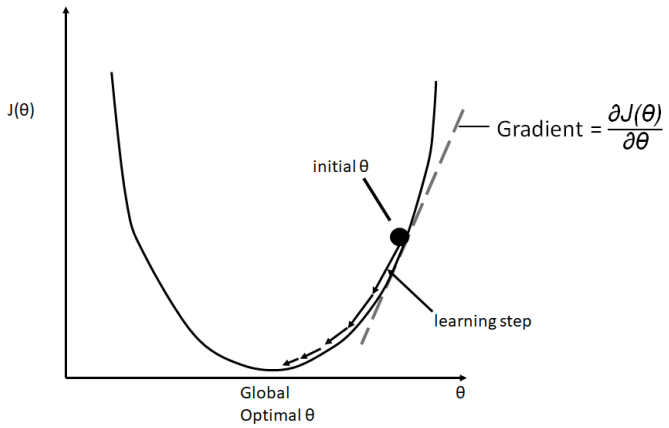


Figure 4.16: Gradient descent algorithm (adapted from [278])

Starting from a random a population of initial weights, θ, it measures the local gradient of θ and takes steps towards the direction of the descending gradient to reduce $J(\theta)$ [278]. The size of these steps moderate the weight change, and they are determined by the *learning rate* (α) parameter [279], according to Equation 4.47. A small learning rate will slow the convergence and cause the algorithm go

through too many iterations, and a large learning rate can cause the algorithm jump in the wrong direction, overshooting the minimum and failing to converge [278]. The user needs to consider these cases and adjust the learning rate manually. A common approach is to experiment with different values and adjust the rates based on the results; however, there are some techniques for adapting learning rates to improve optimisation [280].

$$\theta = \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}) \tag{4.47}$$

One of the challenges of using ANNs is selecting the right *hyperparameters* to get the best from the network. The network learns the weight and biases through training, but the user needs to select the hyperparameters for the network architecture (network layers, number of neurons), loss function, gradient descent optimisation algorithms, and the parameters for the algorithm such as the learning rate, decay, epochs and batch sizes. As neural network systems are becoming increasingly complicated, they are utilising more hyperparameters, and the tuning process is becoming more challenging.

The learning process is an iterative process that involves making m iterations over the entire training data. Iterating over the training data is called an *epoch*. Lighter-weight variants of gradient descent algorithms which operate by picking some random instances of the training data to compute the gradient are more efficient than calculating the gradient over the entire dataset. The *batch size* is the number of training examples utilised in one iteration of the algorithm to compute the gradient; in the case of, stochastic gradient descent (SGD) [279], the batch size is 1. In the case of *mini-batch gradient descent* [279], the batch size is more than one instance and less than the size of the training dataset, usually 32-512. Some of the most widely used gradient descent optimisation algorithms [279] include *Adam*, *RMSprop*, *Adagrad AdaMax* and *Nadam*. Another parameter often used with these optimisers is the *decay* (learning rate schedules) [281], which can be used to further control the learning rate of the algorithm by reducing the learning rate after a set time, e.g. after a defined number of training epochs.

4.4.5 Recurrent Neural Networks

One of the shortcomings of traditional artificial neural networks such as feedforward neural networks (FFNs) was that they were unable to learn from previous data samples to predict the current data sample. Recurrent Neural Networks (RNNs) were introduced to address this shortcoming and for modelling sequential data. The structure of a RNN is similar to that of a standard multilayer FFN but includes connections pointing backwards to allowing it to maintain information about the past. Figure 4.18 illustrates the simplified functionality of an RNN composed of a single neuron, receiving inputs, computing an output, and feeding the output back to itself [264]. At each time step t, it receives the inputs x_t and its own output from the previous time step, y_{t-1}. In this way, the output y_t depends on the current input and output of the previous steps.

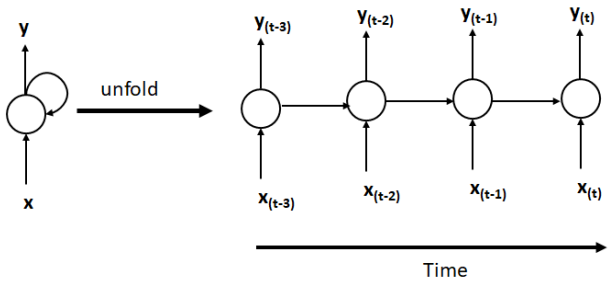


Figure 4.17: A recurrent neuron (left), unfolding in time (right) [264]

Unfortunately, properly training RNNs is difficult due to *vanishing gradient* and *exploding gradient problems* [282]. In these problems, gradient values become extremely small (vanishing gradient) or large errors accumulate (exploding gradients), as a result of which effective learning is significantly impaired. Long short-term memory networks [283] were introduced in 1997 to overcome these problems and produce solutions in which the gradient can flow for longer durations. Since then, they have become of the most effective sequence models, and have been refined and applied to many complex tasks, including time series prediction, robot control, music composition and language translation. In this thesis, we investigate LSTM and another variant of RNN, GRU as time series prediction models to compare how well they learn the normal time sequences of the TE process and so detect

attack data that differs from this.

4.4.5.1 Long-Short Term Memory Network

LSTM [283, 284] overcomes the vanishing gradient and exploding gradient problems of the usual RNN networks by using a complex memory *cell* and four neural network layers with gating controllers to control the flow of information. These cells can be connected recurrently to each other to create a layer of hidden units of recurrent neurons. Figure 4.18, (image taken from [284]), shows the architecture of an LSTM *cell*. The pink circles represent vector multiplication and addition, and yellow lines are learned neural network layers. LSTM maintains two memory states: a long term state, $\mathbf{c}_t$ and a short term state, $\mathbf{h}_t$ (hidden state).

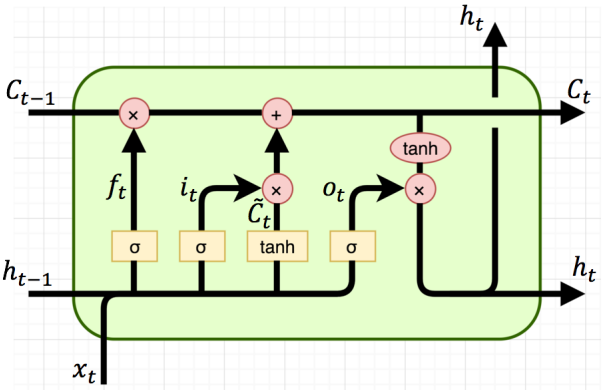


Figure 4.18: LSTM cell, image taken from [284]

The long term state $\mathbf{c}_{t-1}$ runs from left to right [264, 284]. The first step is to decide what to forget. To decide this, the input first goes through the sigmoid layer, also called the *forget gate* f_t, which decides which instances in the current input vector x_t and the previous short term-state h_{t-1} should be removed. Note that the output of the sigmoid functions that are used in three gates ranges from 0-1. If they output a 0, the gate is completely closed, and if the output is 1, it is completely open. Next, the cell decides what new information should be added to the long term state. This is done by the *tanh* layer on the diagram, which analyses the current inputs x_t and the candidates from the previous short term state (h_{t-1}) and creates a vector of new candidates $\tilde{c}_t$ that can be added to the state. The *input gate* (i_t) layer decides what part of the $\tilde{c}_t$ is to be transferred to the long term state. After this,

the new long term state, c_t is obtained. Lastly, the *output gate* (o_t) decides what the next short-term state (h_t) should be, by first passing the previous h_{t-1} and x_t through the sigmoid function, and then multiplying the result with the results obtained from passing the c_t through the tanh.

This process is expressed by the following equations [264]:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
\tilde{c}_t &= \tanh(W_{x\tilde{c}}x_t + W_{h\tilde{c}}h_{t-1} + b_{\tilde{c}}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

where W_{x*} are weight matrices for the four layers and W_{h*} are the weight matrices for their connection to h_{t+1} and b_* are the bias terms for the layers.

4.4.5.2 Gated Recurrent Unit Network

The Gated Recurrent Unit (GRU) [285] is a simplified version of an LSTM, that has become popular over the years due to its simplicity and faster training time.

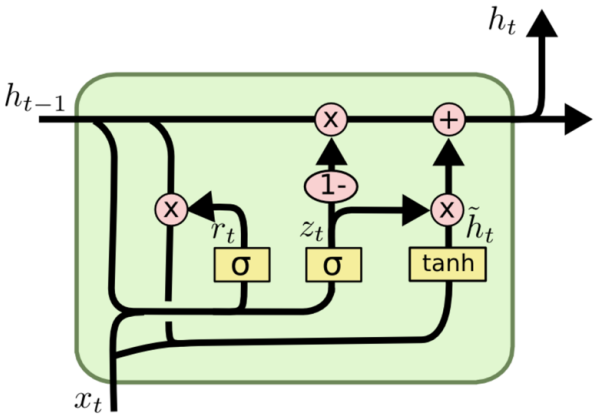


Figure 4.19: GRU cell, image taken from [284]

The differences are illustrated in Figure 4.19 (image taken from [284]). In

GRU, the cell state and hidden state are merged into a single state h_t; GRU also uses a single gate controller, forget and input gate are combined into an update gate [284]. The following equation summarises the operation of the cell for a single instance [264]:

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1})$$
$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1})$$
$$\tilde{h}_t = \tanh(W_{x\tilde{h}}x_t + W_{h\tilde{h}}(r_t \odot h_(t-1)))$$
$$h_t = (1-z_t) \odot tanh(W_{x\tilde{h}}h_{t-1} + z_t \odot \tilde{h}_t)$$

4.4.6 Autoencoder Neural Networks

Autoencoder neural networks are unsupervised learning algorithms that are capable of learning an efficient representation of their inputs from unlabelled data [258, 264]. Autoencoders have been used for the purpose of data compression, dimensionality reduction and, more recently, to facilitate feature learning for unsupervised pretraining in deep learning tasks, and in generative models.

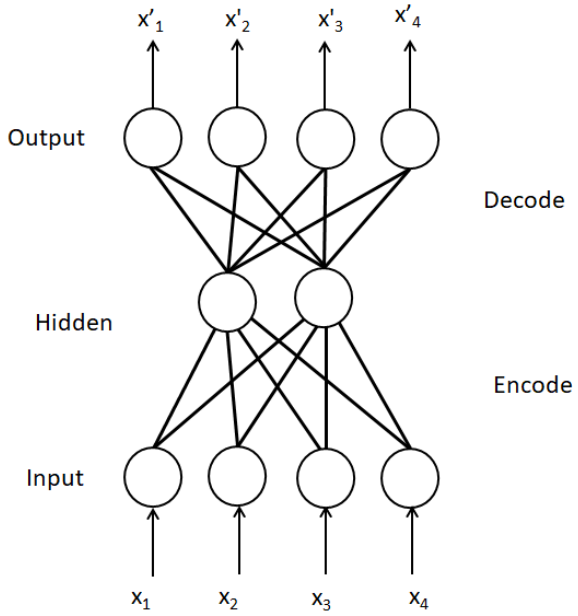


Figure 4.20: A single hidden layer autoencoder neural network

Figure 4.20 illustrates a simple case of an autoencoder composed of three lay-

ers: an input layer, an encoder (hidden) layer and a decoder layer. This is a simple form of a one-hidden-layer FFN; however, the goal of the autoencoder is not to learn a trivial identity function, but to learn $h_{W,b}(x) \approx x$ [286]. That is, to learn an useful approximation of the data, so that the output x' is similar to input x. Therefore, as illustrated in Figure 4.20, the number of the neurons in the output layer must be equal to that for the input. The input, x, is introduced into the network using the input layer. In the encoding process, the hidden layer takes the input and maps it to a hidden representation, h , (called the *code* or latent representation) using weights and biases, and applying the activation function, $h = f(W_x + b)$. Then, in the decoding process, the autoencoder maps h back to the original format to obtain a reconstruction $x' = f(W'_h + b')$. Autoencoders are trained using the backpropagation algorithm, applying the "chain rule to backpropagate error derivatives first through the decoder and then the encoder layer" [258]. The goal of the algorithm is to learn the optimal set of weights and biases by minimising the reconstruction error between x and x'. By imposing constraints on the network, such as reducing the number of neurons in the hidden layer as shown in Figure 4.20, it is possible to prevent the autoencoder from trivially learning the exact mapping from the input to ouput, forcing the autoencoder to learn important features about the data [286]. There are a number of autoenencoders, including *stacked autoencoders*, *variational autoencoders*, *denoising autoencoders*, and *sparse autoencoders*. Autoencoders with multiple hidden layers are called *stacked autoencoders*, and these provide a greater representative power to help in learning complex datasets. Reducing the number of neurons in the hidden layer is one way to constrain the network, but it is not always necessary. It is also possible to use a large number of neurons in the hidden layers, adding a *sparsity* constraint to discover useful structures in the data. The *sparsity* is defined as the average activation of hidden neurons averaged over the training set [286]. Sparsity can be added as penalty term to the cost function of the optimisation to penalise those with high average activation values. Another way to force the autoencoder to learn interesting structures about the data is to feed noise to it. This is the principle behind *denoising autoencoders*, which are given data that

contains corrupting noise to prevent the autoencoder from just learning the identity function, forcing it to learn the most important features of the input (i.e. original uncorrupted data) [257]. Another popular autoencoder, which was discovered in 2013 [287], is the *variational autonencoder* which is sometimes called a *generative autoencoder* because it is used to generate new instances of the data that look like they were taken from the same distribution [264].

Given the capabilities of the autoencoders, we decided to investigate if they can be used as an unsupervised learning model trained on normal operational data to detect attacks. The assumption is that because the autoencoders are not trained on the attack data, they will produce a higher reconstruction error for these cases. Choosing a threshold based on the normal reconstruction error, one could determine if an attack is taking place.

4.5 Summary

In this chapter, we introduced the benchmark Tennessee Eastman process control model to investigate our research questions. The model represents a complex non-linear real-world plant-wide industrial process, and is widely used in control systems research. More recently, it has received some attention from the security community for ICS security studies. We selected an open source reputable MATLAB/Simulink implementation of the model, developed by Ricker [124] containing multiple control loops, 16 process variables and 9 manipulated variables. The model was further extended by Isakov and Krotofil [211] with Simulink blocks to enable attacks to be carried out against the process variable measurements and manipulated variables. We assume an adversary that is capable of carrying out DoS, IntegrityMin, IntegrityMax and replay attacks against these signals. Adversarial motives included causing damage to the safety of the system, and causing economic damage by increasing the operating cost of the plant or damaging the quality of the product.

To identify the most vulnerable components of the process, we specified the problem as a multiobjective optimisation problem, and selected two Pareto-based evolutionary multiobjective algorithms, NSGA-II and SPEA2, for further explo-

ration. In Chapter 5, we develop single attacks against the sensors and actuators in the TE process, and analyse their impacts. In Chapter 6, we use random search, single objective genetic algorithms, and the evolutionary multiobjective algorithms to search the attack space effectively and efficiently to identify the most vulnerable components of the process.

Based on the analysis of literature, we identified a set of classic machine learning models (single Decision Tree, Random Forest, Bagging, AdaBoost, SVM, One-Class SVM), and three deep learning techniques (LSTM, GRU and autoencoder) as the basis for supervised and unsupervised detection models to detect attacks. We discussed their behaviour and key hyperparameters that are important for learning. These models are explored in Chapter 7. Detection models are also vulnerable to attacks, and it is important to identify these vulnerabilities before deployment. As a result, we study detector vulnerabilities by evolving attacks against the detection models using the developed evolutionary multiobjective optimisation approach in Chapter 8.

Chapter 5

Developing Attacks and Investigating their Potential Impacts

In this chapter, we investigate Research Question 1: Can one generate a range of attacks that have a range of consequences? To explore this, individual attacks were carried out against the sensors and actuators of the TE process control model. In the TE model, there are, in total, 25 sensors (measuring process variables) and actuators (receiving manipulated variables) that can be attacked, and for our scenario of attackers who lack control expertise, there are four types of attacks we explored: integrity minimum (*IntegrityMin*), integrity maximum (*IntegrityMax*), *DoS* and *replay* attacks. The DoS and replay attacks do not involve any preparation prior to launch; however, the integrity attacks do require it.

When crafting an integrity attack, an attacker may seek to modify values so that they still lie within the ranges of possible plant signals, but possess the potential to cause some damage, for example to put the plant into an unsafe mode, or increase the operating cost of the plant. To achieve this, attackers first need to listen to the transmitted signals and learn the operating ranges. In the following sections, we carry out experiments to achieve this. Once these parameters are learned, we study the impact of all four attacks on the sensors and actuators to answer Research Question 1.

5.1 Normal Operating Ranges

To carry out the IntegrityMin and IntegrityMax attacks, experiments were conducted to determine the minimum and maximum values of the sensors and actuators observed under normal operating conditions. The plant was operated using operating Mode 1 without attacks and disturbances (the most commonly used configuration in the literature, with a 50/50 product ratio between components G and H) [288], to determine the minimum and maximum values for each of the XMEAS and XMV variables. To provide a credible distribution, 1000 replicates were used with independent random number seeds. The rationale for investigating these variables is to determine the scale of the damage caused by adversaries that use this knowledge to devise attacks. Table 5.1 shows the ranges for sensor (XMEAS) signals, and Table 5.2 illustrates the values for the actuator (XMV) signals. XMV 5, 9 and 12 are constants because they are not updated as part of the control strategy, and thus attacks are not carried out against these actuators. It is worth noting that, although actuators such as these valves take values in the ranges of 0-100 (valves fully open/closed), as shown in the observed values (Table 5.1 and Table 5.2) the full range is not used in practice. This is often the case in the industry; to protect against wear or stiction, valves in plants are never fully closed or opened. The values observed from these experiments were used to devise the IntegrityMin and IntegrityMax attacks described below.

5.1.1 Normal Process Operating Cost

To be able to compare the cost of the attacks with the cost of normal operating, 1000 runs were executed without disturbances, and later in the presence IDV 8 disturbances. This causes random variation in A, B, C feed composition (stream 4), and generates large variations in reactor pressure that better reflect the stochastic nature of a real plant. The following operating costs were obtained:

Operating Mode($)	Max ($)	Mean ($)
Normal	8218.80	8208.02
Normal (with Disturbance IDV8)	8932.97	8305.27

Variable	Variable Number	Min	Max	Mean	Units
A Feed (stream 1)	XMEAS 1	0.25	0.27	0.26	kscmh
D Feed (stream 2)	XMEAS 2	3579.20	3744.46	3662.18	kgh^{-1}
E Feed (stream 3)	XMEAS 3	4339.06	4536.27	4436.62	kgh^{-1}
A and C Feed (stream 4)	XMEAS 4	8.98	9.48	9.24	kscmh
Recycle Flow (stream 8)	XMEAS 5	31.32	33.13	32.20	kscmch
Reactor pressure	XMEAS 7	2793.54	2806.12	2800.00	kPa gauge
Reactor level	XMEAS 8	62.77	67.24	65.00	%
Reactor temperature	XMEAS 9	122.85	122.95	122.90	°C
Purge rate (stream 9)	XMEAS 10	0.1545	0.2689	0.2088	kscmch
Product separator temperature	XMEAS 11	91.46	92.12	91.77	°C
Product separator level	XMEAS 12	45.28	54.74	50.00	%
Product separator underflow (stream 10)	XMEAS 14	24.76	25.92	25.35	$m^3\ h^{-1}$
Stripper level	XMEAS 15	45.29	54.57	50.00	%
Stripper underflow (stream 11)	XMEAS 17	22.37	23.41	22.89	$m^3\ h^{-1}$
Component C (Purge Gas Analysis stream 9)	XMEAS 31	11.87	14.22	13.10	mol %
Component G (Product Analysis stream 11)	XMEAS 40	51.64	55.89	53.88	mol %

Table 5.1: Observed XMEAS signals under normal operating conditions (1000 runs)

Variable	Variable Number	Min	Max	Mean	Units
D feed flow (stream 2)	XMV 1	62.89	63.12	63.02	$kgh^{-}1$
E feed flow (stream 3)	XMV 2	52.99	53.24	53.11	$kgh^{-}1$
A feed flow (stream 1)	XMV 3	25.12	27.045	26.11	kscmh
A and C feed flow (stream 4)	XMV 4	59.93	61.32	60.57	kscmh
Compressor recycle valve	XMV 5	0.0	0.0	0.0	%
Purge Valve (stream 9)	XMV 6	19.39	32.76	25.71	%
Separator pot liquid flow (stream 10)	XMV 7	37.21	37.46	37.33	$m^3\ h^{-}1$
Stripper liquid product flow (stream 11)	XMV 8	46.36	46.55	46.46	$m^3\ h^{-}1$
Stripper steam valve	XMV 9	0.0	0.0	0.0	%
Reactor cooling water flow	XMV 10	35.46	36.33	35.90	$m^3\ h^{-}1$
Condenser cooling water flow	XMV 11	5.20	19.69	12.52	$m^3\ h^{-}1$
Agitator Speed	XMV 12	100.00	100.00	100.00	rmp

Table 5.2: Observed XMV signals under normal operating conditions (1000 runs)

As expected with disturbances, the mean value is slightly higher, and it reaches a maximum value of $8932.97. This shows that the control system is able to compensate for the disturbances smoothly without causing large production rate changes and significantly increased overall operating costs. Both from the attacker's perspective and plant owners' perspective these values are important benchmarks. An attacker needs to ensure that the investment in attacks, such as buying vulnerabilities to exploit, developing skills and other preparation can be justified by the consequences of the attack, assuming their intended impact is economic damage. Similarly, the plant operators may only mitigate vulnerabilities if they can justify the downtime of the plant and other costs relating to patching of these vulnerabilities.

We decided not to use disturbances in the presence of attacks because they make the job of the detection harder: the detection will need to recognise the difference between normal operating, disturbances and the attacks. This is beneficial to the attacker and is particularly vulnerable when generating new attacks using evolutionary computation, since the detector's performance will be less accurate. The variations in the disturbances may also improve the impact of the attack, such as causing the plant to shut down faster, and increase the operating cost. We opted to make the generation of new attacks harder, and decided not to use the disturbances. This also makes the comparison of our detection mechanisms easier. The existing intrusion detection mechanisms proposed in the literature for the TE process control model focus on detecting attacks without the disturbances, and the fault diagnosis mechanisms are designed to distinguish between normal operation and disturbances. The generation of attacks in the presence of disturbances is therefore left as future work.

5.2 Single Random Attacks

As mentioned above, the TE plant was operated using operating Mode 1, and all subsequent experiments were carried out using this mode. Before investigating the random strategy and techniques from evolutionary computation to search for op-

timal combinations of attacks, we carried out 500 random attacks on each of the sensors and actuators to determine the consequences of single attacks. We later, in Chapter 6, use this knowledge as a comparison for the results obtained from the random strategy and evolutionary computation algorithms. The attacks can be carried out using:

- interval mode: attacks start from the time of the attack and continue for the given duration.
- periodic mode: attacks start from the starting time of the attack, and last until the duration of the attack is complete. However, this is done periodically, as defined by setting a pulse (wait) period.

As expected, periodic attacks were slower and required more elapsed time to have a significant effect. Because the aim of this work was to understand the particular vulnerabilities of the system by maximising damage, we elected to focus on the interval attacks and report the results of this in the following subsections.

5.2.1 Impact of Attacking Process Variable Measurements

For each of the process variables (XMEAS), integrity attacks were carried out using the maximum and minimum values obtained from the normal execution, together with DoS attacks and replay attacks. 500 attacks were carried out on each sensor, each started at hour 2 and was intended to run for the rest of the 72 hours simulation time (i.e. a duration of 70 hours). The impact of integrity attacks on the sensors is shown in Table 5.3 and the impact of the DoS and replay attacks on the sensors is shown in Table 5.4. If the plant exceeds the safety limits, it shuts down, and a warning message is displayed to explain the reason for the shut down. The plant automatically shuts down as a result of: *Low/High Reactor pressure, High Reactor Temperature Low/High Product Separator Level* and *Low/High Stripper Base Level*, as described in [20]. The *shutdowns* column denotes the total number of times the plant shut down as a consequence of the attack out of 500 runs. The *shutdown range* column indicates the time at which the plant shuts down after the attack was started.

XMEAS	Attack	Max Cost	Mean Cost	Shutdowns	Shutdown Range(Hrs)
(1) A-Feed	Max	8449	8407	0	-
	Min	8364	8260	0	-
(2) D Feed	Max	1158	1152	500	3.43-3.48
	Min	915	902	500	4.31-4.4
(3) E Feed	Max	739	730	500	2.66-2.72
	Min	1320	1312	500	4.17-4.22
(4) A and C Feed	Max	384	378	500	1.25-1.34
	Min	392	361	500	**0.52-0.65**
(5) Recycle Flow	Max	2099	2040	500	8.15-8.42
	Min	3456	3415	500	10.58-10.78
(7) Reactor pressure	Max	**24507**	24468	0	-
	Min	671	650	500	8.37-8.78
(8) Reactor level	Max	381	375	500	2.81-2.87
	Min	1017	1008	500	2.83-2.89
(9) Reactor temperature	Max	364	356	500	**0.59-0.63**
	Min	377	366	500	1.23-1.28
(10) Purge rate	Max	8228	8195	0	-
	Min	8246	8218	0	-
(11) Separator temperature	Max	**10427**	10364	0	-
	Min	10444	10358	0	-
(12) Separator level	Max	896	888	500	5.85-5.95
	Min	1256	1245	500	8.57-8.68
(14) Separator underflow	Max	1370	1357	500	9.09-9.79
	Min	1449	1385	500	9.71-10.41
(15) Stripper level	Max	1756	1740	500	13.48-13.71
	Min	1804	1793	500	13.31-13.54
(17) Stripper underflow	Max	312	305	500	**1.02-1.03**
	Min	387	379	500	1.01-1.02
(31) C in Purge	Max	**20515**	20438	500	65.65-66.1
	Min	13493	13455	500	50.02-50.4
(40) G in product	Max	8312	8298	0	-
	Min	8117	8106	0	-

Table 5.3: Impact of IntegrityMin and IntegrityMax attacks on XMEAS signals (500 Runs)

XMEAS	Attack	Max Cost	Mean Cost	Shutdowns	Shutdown Range(Hrs)
(1) A-Feed	DoS	8447	8331	0	-
	Replay	8429	8316	0	-
(2) D Feed	DoS	3149	1761	500	6.9-21.86
	Replay	3897	2765	500	13.64-30.42
(3) E Feed	DoS	2480	1494	500	3.57-18.16
	Replay	3350	2330	500	11.65-22.9
(4) A and C Feed	DoS	1318	743	500	1.38-6.48
	Replay	1227	825	500	2.32-5.99
(5) Recycle Flow	DoS	21942	10675	322	10.4-69.7
	Replay	**22429**	9169	3	64.67-64.67
(7) Reactor pressure	DoS	**24299**	10430	254	9.01-60.3
	Replay	23889	10894	233	9.73-66.15
(8) Reactor level	DoS	7435	1989	494	3.77-35.79
	Replay	**11302**	5058	418	15.67-67.34
(9) Reactor temperature	DoS	10464	1554	489	0.92-61.64
	Replay	**10774**	3681	467	2.35-67.04
(10) Purge rate	DoS	8235	8201	0	-
	Replay	8236	8201	0	-
(11) Separator temperature	DoS	10386	10270	0	-
	Replay	**10393**	10264	0	-
(12) Separator level	DoS	8210	3816	463	8.38-66.43
	Replay	8217	7802	143	39.66-69.96
(14) Separator underflow	DoS	4038	2274	500	11.07-32.98
	Replay	4431	3004	500	19.17-36.51
(15) Stripper level	DoS	8234	5368	413	19.77-69.61
	Replay	8234	8181	16	57.03-68.13
(17) Stripper underflow	DoS	4512	2298	500	6.68-37.75
	Replay	5716	4312	500	28.45-48.24
(31) C in Purge	DoS	**17205**	9841	0	-
	Replay	10951	8818	0	-
(40) G in product	DoS	8267	8208	0	-
	Replay	8268	8212	0	-

Table 5.4: Impact of DoS and replay attacks on XMEAS signals (500 Runs)

5.2.1.1 Impact of Attacks on Plant Safety

As shown in Table 5.3, most of the integrity attacks on the sensors caused the plant to shut down. Integrity attacks are able to bring down the plant faster than the DoS and replay attacks shown in Table 5.4. An adversary that wants to bring down the plant in the shortest time may want to choose to attack the sensor that achieve this. This is necessary to give the defender less time to react. Our experiments show that the fastest attack that can bring the plant down are an IntegrityMin attack on A and C Feed Flow (XMEAS 4), requiring an attack to last for a duration of 0.52-0.65 hours (31.2-39 minutes), and an IntegrityMax attack on Reactor Temperature (XMEAS 9), resulting in a shut down at between 0.59-0.63 hours (35.4-37.8 minutes). In these cases, the controller receives fake values from XMEAS 4 and XMEAS 9, which are lower for XMEAS 4 and higher for XMEAS 9 than the legitimate values, and calculates the control values that are sent to the actuators using these fake values. This behaviour causes a significant increase in the reactor pressure, and the plant shuts down as result.

The integrity attacks on XMEAS 17 (Stripper underflow) on average take just over one hour to shut down the plant. Attacks on other sensors can take anything from 1.2 to 68 hours to achieve the same effect. These are long durations for attacks and make detection likely. What happens after attacks are detected, including whether the operators have enough time to take the necessary countermeasures to bring the plant back to safe operating conditions are important questions; however, they will be the focus of future work. The IntegrityMax attack on Reactor Pressure, and attacks on A-Feed, Purge Rate, Product Separator Temperature and mol% in G were not successful, so these variables do not appear to be good candidates to bring down the plant using the values observed.

In common with previous studies [208], and for reasons for comparability, the objective of the DoS attack is to stop signals reaching the controllers and/or actuators. In the absence of new data, we assume that the controller or the actuator uses the last received signal for the period of the attack. For the replay attack, the data collected earlier in the plant, before hour 2, were replayed later when carrying

out the attacks. The results obtained from these attacks are illustrated in Table 5.4. DoS attacks outperformed replay attacks but they take longer to shut down the plant in comparison to integrity attacks.

5.2.1.2 Impact of Attacks on Economic Cost

In the long run, attacks that could cause economic damage by increasing the operating costs of running the plant can cause the plant owner more damage than by shutting down the plant. The operating cost is determined by considering the loss of product in the purge stream, loss of product in the product stream, the cost of amount of F formed, and the cost of running the compressor and stripper steam, as discussed in Chapter 3. This cost is dominated by the loss of raw materials in purge and in operating the compressor [288]. Rather than define arbitrary metrics ourselves, we use the predefined operating function that calculates the operating cost of the plant to determine the economic damaged caused by the attacker.

As reported in Table 5.3, the experiments carried out showed that the integrity attacks that have the potential to increase the cost of operating the plant are:

- An IntegrityMax attack on Reactor Pressure (XMEAS 7), which increased the operating cost from the average of $8208 to $24507 with no risk of shutting the plant down.

- IntegrityMax and IntegrityMin attacks on the separator temperature (XMEAS 11), which increased the operating cost to $10427-10444 with no risk of shutting the plant down.

- An IntegrityMax attack on the sensor measuring Component C in Purge (XMEAS 31), which increased the operating cost to $20515; however, one needs run this attack for a duration of less than 65 hours, as longer attacks cause the plant to shut down.

The economic impact of the DoS attacks and replay attacks are reported in Table 5.4. Some of the attacks that increase the operating cost of the plant significantly are:

- A DoS attack on Reactor Pressure (XMEAS 7), which increased the operating cost to $24299 with 50% risk of shutting the plant down.
- A replay attack on Recycle Flow (XMEAS 5), which increased the operating cost to $22429 with low (0.4%) risk of shutting the plant down.
- A DoS attack on C in Purge (XMEAS 31), which increased the operating cost to $17205 with no risk of shutting the plant down.
- A DoS or a replay attack on Separator Temperature (XMEAS 11), which increased the operating cost to $10386-10393 with no risk of shutting the plant down.

If attacks are not designed carefully, they have the potential to shut down the plant. To avoid this risk, the attacker will need to decide the best time to attack in real time, or choose a less effective attack. For example, carrying out DoS or replay attacks on Separator Temperature (XMEAS 11) has the potential to increase the cost to just over $10380, and, a DoS attack on Component C in Purge (XMEAS 31), has the potential to increase the cost to $17205; both avoid shutting down the plant.

5.2.1.3 Impact of Attacks on Quality of the Product

None of the attacks carried out had a harmful effect on product quality.

5.2.2 Impact of Attacking Manipulated Variables

As with XMEAS, to assess the impact of the attacking the manipulated values issued by the controller to actuators, 500 attacks were launched against the plant for each attack type and against each actuator. Attacks were started at hour 2 with the intention of running them for the remainder of the simulation time (i.e. 70 hours).

XMV	Attack	Max Cost	Mean Cost	Shutdowns	Shutdown Range(Hrs)
(1) D Feed	Max	8209	8198	0	-
	Min	8226	8217	0	-
(2) E Feed	Max	8234	8221	0	-
	Min	8204	8194	0	-
(3) A Feed	Max	8352	8342	0	-
	Min	8259	8248	0	-
(4) A and C Feed	Max	**10101**	10085	0	-
	Min	9339	9290	500	38.49-39.08
(6) Purge Valve	Max	**9552**	9542	0	-
	Min	7223	7215	0	-
(7) Separator Pot liquid	Max	2382	2313	500	17.65-18.92
	Min	3355	3273	500	25.83-27.35
(8) Stripper Liquid	Max	2014	1962	500	15.18-15.99
	Min	2097	2060	500	15.38-16.11
(10) Reactor Cooling	Max	786	701	500	1.65-2.01
	Min	**11703**	6651	489	18.88-67.49
(11) Condenser Cooling	Max	325	319	500	1.59-1.6
	Min	414	408	500	**0.64-0.65**

Table 5.5: Impact of IntegrityMin and IntegrityMax attacks on XMV signals (500 Runs)

The impact of these attacks is reported in Table 5.5 and Table 5.6. Please note that attacks that were not carried on actuators XMV 5, XMV 9 and XMV 12 and these are missing from Table 5.5 and Table 5.6 because these have constant values and their positions are not modified by the simulated control strategy.

5.2.2.1 Impact of Attacks on Plant Safety

As reported in Table 5.5, the model is resilient to integrity attacks on valves controlling the D feed flow (XMV 1), E feed flow (XMV 2) and A feed flow (XMV 3) and Purge Valve (XMV6), as these do not seem to damage plant safety. The remainder of the actuators are not resilient to integrity attacks. The attack that caused the fastest damage is the IntegrityMin attack on the Condenser Cooling water flow

XMV	Attack	Max Cost	Mean Cost	Shutdowns	Shutdown Range (Hrs)
(1) D Feed	DoS	8216	8204	0	-
	Replay	**9347**	8223	0	-
(2) E Feed	DoS	8215	8204	0	-
	Replay	8216	8203	0	-
(3) A Feed	DoS	8277	8216	0	-
	Replay	8247	8210	0	-
(4) A and C Feed	DoS	**14972**	8433	2	68.09-68.09
	Replay	8873	8248	0	-
(6) Purge Valve	DoS	**9064**	8221	0	-
	Replay	8876	8234	0	-
(7) Separator Pot liquid	DoS	8217	7578	131	26.01-69.42
	Replay	8217	7975	82	44.05-68.92
(8) Stripper Liquid	DoS	8231	5698	366	22.32-68.72
	Replay	8235	6387	356	24.54-69.52
(10) Reactor Cooling	DoS	6093	1976	500	6.18-34.73
	Replay	5909	2283	500	6.4-34.61
(11) Condenser Cooling	DoS	10803	2268	477	1.61-54.62
	Replay	**11596**	7782	108	38.32-69.49

Table 5.6: Impact of DoS and replay attacks on XMV signals (500 Runs)

(XMV 11), resulting in a shut down time of 0.64 hours (38.4 minutes) due to low separator liquid level. Carrying out a IntegrityMax attack on Reactor cooling water flow (XMV 10) can shut down the plant in 1.65 hours (99 minutes) due to high reactor pressure. A and C feed flow (XMV 4), Separator pot liquid flow (XMV 7) and Stripper liquid product flow (XMV 8) are also vulnerable to shut down attacks, causing the plant to shut down due to high pressure, low/high stripper or low/high separator liquid level. As shown in Table 5.5, these attacks require more time to shut down the plant. The impact of the DoS and replay attacks showed similar performance to integrity attacks; however, overall they require more time to cause the plant to shut down as illustrated in Table 5.6.

5.2.2.2 Impact of Attacks on Economic Cost

Integrity attacks on A and C feed flow (XMV 4), Purge valve (XMV 6) and Reactor Cooling water flow valve (XMV 10) have the potential to increase the operating cost; however, the effect is smaller than the attacks on XMEAS, as shown in Table 5.5:

- An IntegrityMax attack on A and C Feed valve (XMV 4), which increased the operating cost from the average of $8208 to $10101 with no risk of shutting the plant down.

- An IntegirtyMin attack on the Reactor Cooling valve (XMV 10), which increased the operating cost to $11703 but has a high risk of shutting the plant down, at 97.8% (489/500).

- An IntegrityMax attack on the Purge valve (XMV 6), which increased the operating cost to $9552 with no risk of shutting the plant down.

The economic impact of the DoS attacks and replay attacks are for XMVs are reported in Table 5.6. Some of the attacks that increase the operating cost of the plant are:

- A DoS attack on A and C Feed valve (XMV 4), which increased the operating cost to $14972 with low risk of shutting the plant down.

- A DoS attack on Condenser Cooling valve (XMV 11), which increased the operating cost to $10803 with high risk of shutting the plant down. A replay attack did better than the DoS, increasing the operating cost to $11596, and being less likely to shut down the plant.

- A replay attack on D Feed valve (XMV 1), which increased the operating cost to $9347 with no risk of shutting the plant down.

- DoS and replay attacks on Purge valve (XMV 6) increased the operating cost to $9064-8876 with no risk of shutting the plant down.

5.2.2.3 Impact of Attacks on Quality of the Product

None of the attacks carried out had an harmful effect on the product quality.

5.3 Summary

The results obtained from these experiments answer our Research Question 1: we are able to generate a range of attacks that have consequences for the safety of

the plant and the operating cost. Our results show that attacks against the process variable measurements are capable of causing more damage than attacks against the manipulated variables. Integrity attacks have the potential to cause more significant damage both in terms of economic damage and the time required to shut down the plant. However, the attacker needs to carefully select both the right target and time of the attack to achieve the intended impact, itself a choice between shutting down the plant or causing economic damage without shutting it down. Interestingly, no single attack is able to affect product quality significantly.

Given an affirmative answer to Research Question 1, we next explore Research Question 2 in the following chapter to determine whether it is possible to automate this process and find even better attacks, by developing an effective and efficient search to combinations of attacks.

Chapter 6

Searching for an Effective and Efficient Attack Generation Approach

In the previous chapter, we studied the TE model, implemented a range of attacks and tested single attacks against each sensor and actuator. Our results showed that we can generate a range of attacks with different consequences. Based on these results, in this chapter we investigate Research Question 2 (How should one search the attack space effectively and efficiently, and identify the most vulnerable components of the process?) using the methods we identified in Chapter 4 to design an approach that can be used to study combinations of attacks. To achieve this, we broke Research Question 2 further into two parts:

- Can we search for the attacks that cause the most damage (e.g. increase the operating cost by the greatest amount)?
- Can we optimise these attacks in terms of effort required (number of sensors and actuators attacked) and damage caused (in terms of safety and economic damage) to determine the most vulnerable combinations of devices?

In Section 6.1, we investigate the first part of the question, using a single objective genetic algorithm (GA), and compare the performance of the GA to a random search to determine if evolutionary algorithms are suitable candidate for our prob-

lem. For the comparison study, we decided to keep the search simple by evolving only DoS attacks. In Section 6.1.5, we use the GA to evolve multiple types of attacks to determine the ability of the evolutionary algorithms to identify good candidates in a larger space of attacks. Following this, we chose to investigate if it is possible to optimise attacks using evolutionary multiobjective optimisation (EMO) to identify the vulnerabilities in a complex industrial control system without requiring a detailed mathematical model of it. In Section 6.2, we develop and test the EMOs to generate attacks to damage the safety of the system, and to cause economic damage.

Please note that the word individual and chromosome, which is also known as genotype in other literature, are used interchangeably throughout the Chapter.

6.1 Comparison of Random Search and Genetic Algorithm

In order to measure the effectiveness of using evolutionary algorithms as an attack generator and show their ability to find vulnerabilities, random search was compared to a GA, and both results compared to the single attacks studied in previous chapter. For this part of the study, the two approaches were used to generate DoS attacks aimed at increasing the operating cost of the plant whilst avoiding shut down. The duration of DoS attacks was 70 hours, starting at hour 2. In the following subsections, we discuss some practical challenges we faced, then explain the random generation and GA algorithms used to generate attacks. Finally, we discuss and report results.

6.1.1 Practical Challenges

The TE model is implemented in MATLAB/Simulink. Once the attacks are generated, they need to be executed on the TE model to determine the performance of the attack on the process in terms of the total cost of the operating the plant, the operating length of the plant and the quality of the product. Due to the computationally intensive nature of this work, all experiments were carried out using the

high performance computing facilities at UCL. The computing platform used for all experiments throughout this chapter are cluster nodes running 2.00GHz Intel Xeon E5-2620 CPUs under CentOS release 6.5.

One of the key challenges running experiments has been the execution time of the MATLAB/Simulink TE model on the high performance computing facilities. Leaving the MATLAB startup overhead aside, running a single model could take up to 25 seconds, and, for some attacks such as replay attacks, it can take longer. Running an evolutionary algorithm, with a population of 200 individuals for 200 generations, may require 40,000 evaluations. This means run time of the evaluation can take 278 hours or 11.6 days on a single machine, excluding the additional overhead of the generation of the attacks. To improve the performance, multiple individuals, MATLAB jobs were run to evaluate the fitness of the individuals in parallel, and the genetic algorithm library we used was extended to collect the fitness of each attack from these jobs. The main program running the random search or the evolutionary computation distributes the individuals MATLAB jobs; for example, if there are 20 MATLAB compute nodes running, the 200 individuals are distributed among these 20 nodes. The main program then waits for these jobs to be completed, collects the fitness values for the distributed individuals and then generates a new generation of the individuals. It repeats this until the evolutionary algorithm terminates. Given the time and resource constraints, the genetic material of the algorithms describes only a limited set of attack durations and start times.

6.1.1.1 Implementation Library

It is neither ideal nor recommended to write evolutionary algorithms from scratch as there are many good frameworks that have been extensively used and tested. We tested several of these open source frameworks for evolutionary computation. At the beginning of our work, we made a choice to use Python library called Pyevolve [289] for its ease of use and extensible capabilities, based on our previous work [152] [290]. However, this did not offer any support for implementing multiobjective evolutionary algorithms. We found several widely used frameworks supporting multiobjective evolutionary algorithms, including a Java framework, the Evolution-

ary Computation Toolkit (ECJ) [291], and a Python framework, Distributed Evolutionary Algorithms in Python (DEAP) [292]. We opted to use DEAP because it was superior in terms of prototyping: it required fewer lines of code, making the implementation and testing more manageable. Once the attacks are generated using DEAP in Python, they are converted to MATLAB scripts in the main program (also implemented in Python) to be executed on the TE Model.

6.1.2 Generating Attacks using a Single Objective Genetic Algorithm

Table 6.1 shows the parameters chosen for the single objective GA. Algorithm 5 shows the pseudocode for the GA. This GA uses a list of bits to indicate which of the two forms of each gene are encoded. Thus, an individual (chromosome) is a list of binary values, such as [0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0], where the first 16 genes correspond to the 16 sensors (XMEAS) and the remaining 9 genes (XMV) correspond to the actuators. For example, this chromosome corresponds to an attack using the following parameters:
$XMEAS1_{\text{NoAttack}}, XMEAS2_{\text{Noattack}}, XMEAS3_{\text{DoS}}, XMEAS4_{\text{DoS}}, XMEAS5_{\text{DoS}},$
$XMEAS7_{\text{DoS}}, XMEAS8_{\text{DoS}}, XMEAS9_{\text{DoS}}\ XMEAS10_{\text{DoS}}, XMEAS11_{\text{DoS}}, XMEAS12_{\text{DoS}},$
$XMEAS14_{\text{DoS}}, XMEAS15_{\text{NoAttack}}, XMEAS17_{\text{NoAttack}}, XMEAS31_{\text{NoAttack}}, XMEAS40_{\text{DoS}},$
$XMV1_{\text{NoAttack}}, XMV2_{\text{DoS}}, XMV3_{\text{NoAttack}}, XMV4_{\text{NoAttack}}, XMV6_{\text{NoAttack}},$
$XMV7_{\text{NoAttack}}, XMV8_{\text{NoAttack}}, XMV10_{\text{NoAttack}}, XMV11_{\text{NoAttack}}.$

Initial Population. The GA uses a random number generator that takes an integer as a seed and is used to generate the initial population randomly. Each member of the population encodes a possible combination of attacks. The initial population can impact the search and consequent evolution can converge towards different solutions; thus, it is important to run the GA multiple times to study the reproducibility of the results. A different seed was used for each replicate to ensure diversity in the initial population. Each gene of the chromosome is sampled from a uniform Bernoulli distribution.

Parameters	Value
Chromosome size	25
Types of genes	2
Chromosome encoding	Binary
Description of genes	Do not Attack, DoS
Number of generations	200
Parent population size	200
Offspring population size	200
Crossover probability	0.8
Mutation probability	0.2
Probability of mutating a gene in a chromosome	0.05
Crossover operator	Two-point crossover
Mutation operator	Flip-bit
Selection operator	Tournament
Parallel evaluations	20
Computation time	$\approx$ 15 hrs

Table 6.1: Evolutionary operators and parameters for generating DoS attacks using GA

Algorithm 5: Genetic algorithm for generating attacks

Input : $mutpb$ = mutation rate, $cxpb$=crossover rate,
$ngen$ = number of generation
Output: best individual

```
1  pop = generate initial population randomly
2  pop=evaluate(pop)
3  gen=1
4  while gen ≤ ngens do
5      offspring = selTournament(pop)
6      for child1, child2 in offspring do
7          random = randomGenerator(0, 1)
8          if random() < cxpb then
9              child1, child2 = crossover (child1, child2)
10     for child in offspring do
11         if random() < mutpb then
12             child = mutate(child)
13     offspring = evaluate(offspring)
14     pop = offspring
15     gen = gen + 1
16 best=selectBest(pop)
17 return best
```

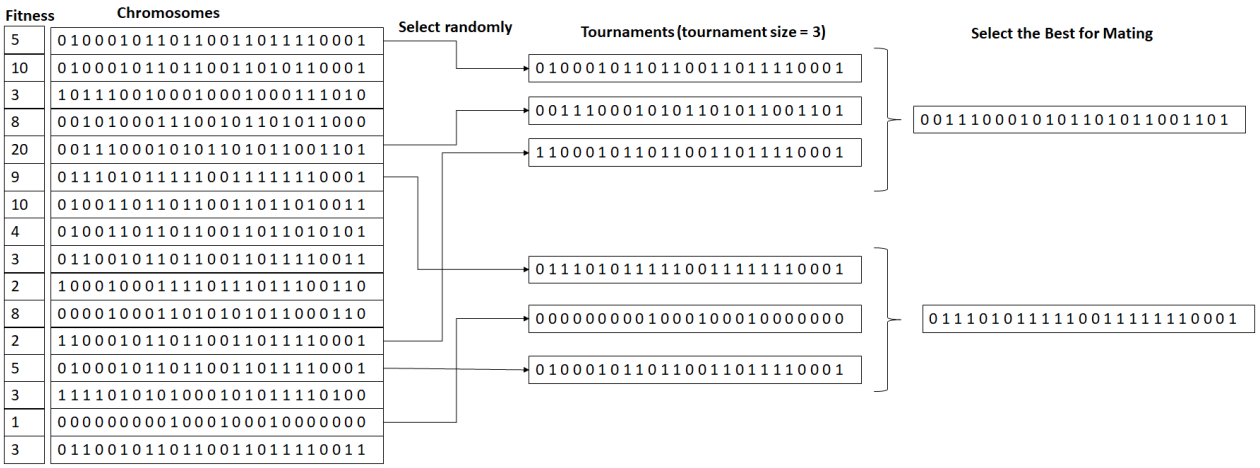
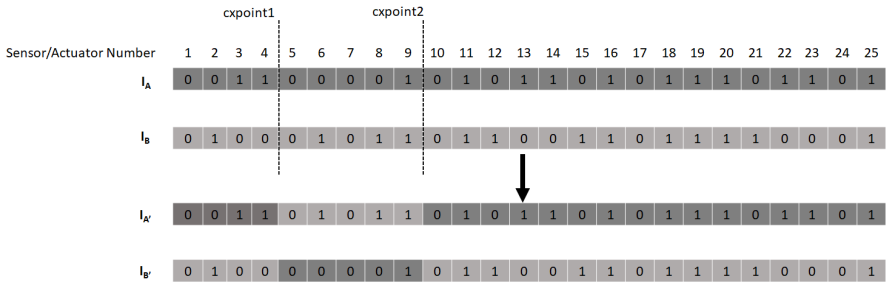
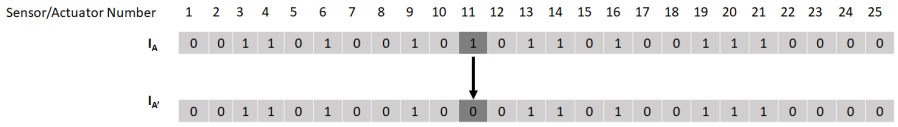


Figure 6.1: The tournament selection process for selecting candidates for mating pool



(a) Two-Point Crossover Operator used on two selected chromosomes



(b) Mutation Operator used on an chromosome

Figure 6.2: Two-point crossover and mutation operation for DoS Attacks

Tournament Selection. The selection method used for the GA is the tournament selection method [293]. This selects the best chromosome among tournament size (*tournsize*) randomly chosen chromosomes. This process is repeated k times, as illustrated in Figure 6.1.

Two-point Crossover. Two-point crossover was used since the chromosomes are composed of binary values. These points are chosen randomly and the relevant genetic material is exchanged, ensuring that chromosomes keep their original length, as illustrated in Figure 6.2a.

Flip-bit Mutation. For this particular GA, in which the chromosomes are

composed of only binary values, flip mutation is used to modify the value of the gene. The number of genes in each chromosome selected for mutation is determined by the selected probability of each mutation which, in this case, is 0.05, as shown in Table 6.1. Figure 6.2b shows application of the mutation operator on a chromosome.

Population and Offspring Size. The number of offspring produced at each generation is the same as the parent population size. At each generation, new offspring are produced using the genetic operators, and the previous population is entirely replaced by the new offspring.

Fitness Function. The goal of the GA is to maximise fitness. In our case, the fitness of a chromosome is measured according to the total operating cost of the plant after converting the chromosome to a MATLAB script and executing the attack on the plant. Thus, the higher the operating cost, the better the chromosome.

Termination Condition. The GA was designed to run for a maximum number of generations. After running the GA for the defined number of generations, the fittest individual in the final population is returned as the best attack strategy.

6.1.3 Generating Attacks using Random Search

The attacks for the random generator are produced in the same way the evolutionary computation generates its initial population. Each individual in the population has a chromosome with 25 genes, and each gene corresponds to an sensor or an actuator. The first 16 genes correspond to the sensors (XMEAS), and the last 9 correspond to the actuators (XMV). Using this approach, 6 sets of 40,000 randomly generated attack strategies were generated and evaluated. The size of the set is much larger than the number of possible attacks that the GA is able to generate throughout its evolution.

6.1.4 Results of Random Search and Genetic Algorithm

The metrics used for the analysing the GA are: the statistics collected at each generation for the maximum, minimum and average fitness; the fittest individual returned by the algorithm; and the number of unique attack solutions obtained.

Histograms showing the fitness of the attacks generated using random search

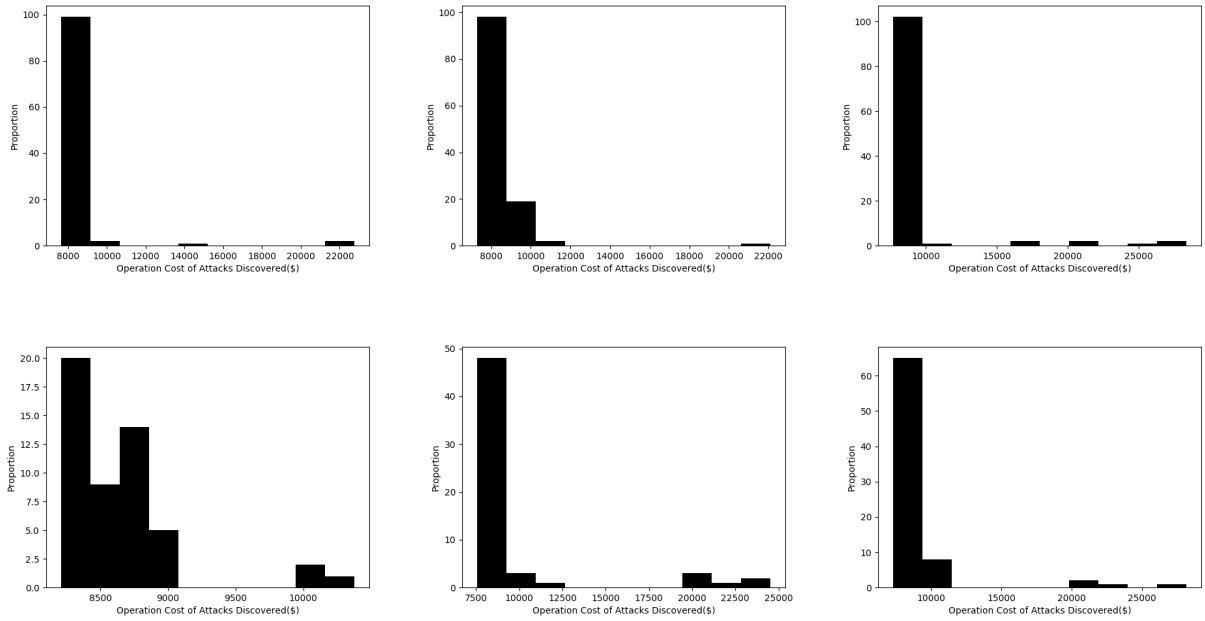


Figure 6.3: Fitness distribution of attacks generated using Random Search

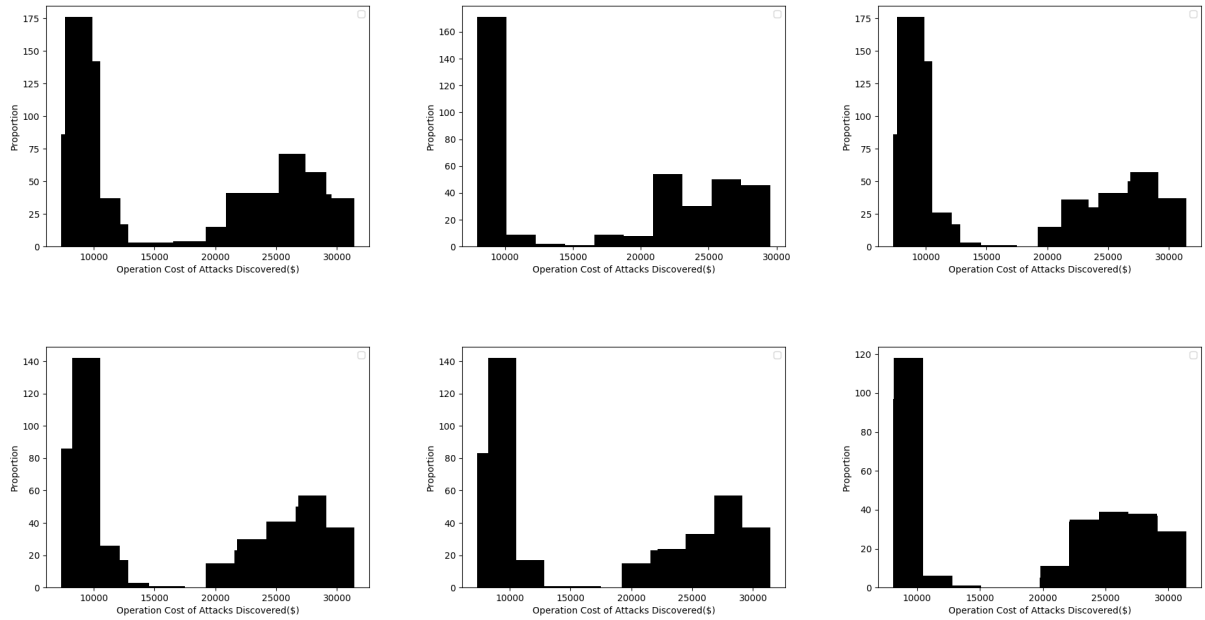


Figure 6.4: Fitness distribution of attacks generated using Genetic Algorithm

are shown in Figure 6.3. The histograms illustrate the unique attacks (duplicates removed) that kept the plant operating while causing economic damage by increasing the operating cost. On average, random search found 86 unique DoS attacks per run, far fewer than the number of attacks generated by the GA, which generated on average 299 unique attacks, as illustrated in Figure 6.4. Only 4 of the attacks generated by random search increased the operating cost (from $8,218 to $20,000), causing a loss in the excess of $11,782. Earlier, in Section 5.2, the maximum operating cost we found using single DoS attacks was the attack on Reactor Pressure (XMEAS 7), which caused the operating cost to increase to $24,299. Random search produced only one attack strategy that was better than the single DoS attack on XMEAS 7, and this increased the cost of the plant operating to $28,148. This involved attacking XMEAS 7 with other sensors and actuators.

The GA was much more effective at generating attacks that kept the plant on and increased the operating cost. The histograms in Figure 6.4 show the number of attacks found, with their operating costs, for 6 runs of the GA. For each run, a new random seed was used for the GA to ensure that the initial population was changed. A different seed was also used for the plant for the each evolution to ensure randomness of the plant.

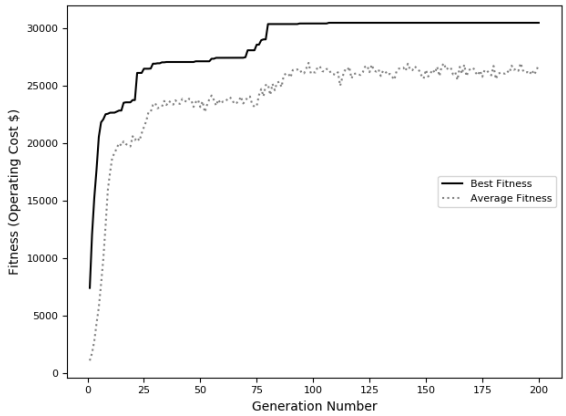
Duplicates, and those attacks that shut down the plant, were removed from the plotted results. Overall, the GA generated more unique attacks, giving a range of attacks producing different operating costs. The best individual returned at the end of the evolution caused more damage than the single attacks carried out on the XMEAS and XMV values and results obtained from the random search, producing attacks with operating cost from $29,553 to $31,449. Across the 6 runs, the GA produced between 196-380 unique attacks. Given that these are long duration attacks, and the probability of their success (i.e. causing damage whilst avoiding plant to shut down) is very low, the GA proved to be an effective and efficient approach for finding the set of feasible attacks. The best attack found during the evolution is shown in Table 6.2.

Figure 6.5 shows the performance of the single objective GA. Figure 6.5a

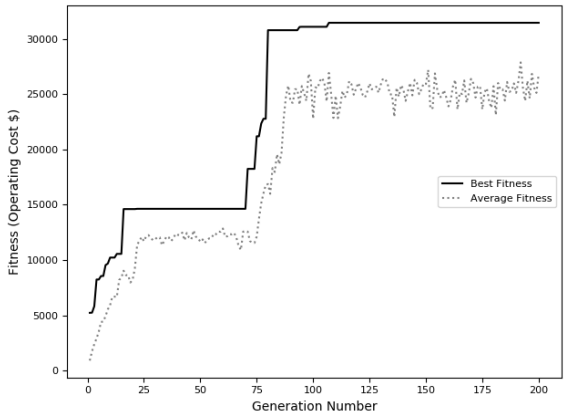
Chromosome	0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0
Description	$XMEAS1_{NoAttack}$,$XMEAS2_{NoAttack}$, $XMEAS3_{NoAttack}$, $XMEAS4_{NoAttack}$,$XMEAS5_{NoAttack}$,$XMEAS7_{DoS}$, $XMEAS8_{NoAttack}$,$XMEAS9_{NoAttack}$, $XMEAS10_{DoS}$, $XMEAS11_{NoAttack}$, $XMEAS12_{NoAttack}$,$XMEAS14_{NoAttack}$, $XMEAS15_{NoAttack}$,$XMEAS17_{NoAttack}$, $XMEAS31_{NoAttack}$, $XMEAS40_{DoS}$, $XMV1_{NoAttack}$,$XMV2_{NoAttack}$, $XMV3_{NoAttack}$, $XMV4_{DoS}$, $XMV6_{NoAttack}$, $XMV7_{NoAttack}$, $XMV8_{NoAttack}$, $XMV10_{DoS}$, $XMV11_{NoAttack}$
Cost ($)	31449.00

Table 6.2: Example of a GA generated attack with high operating cost

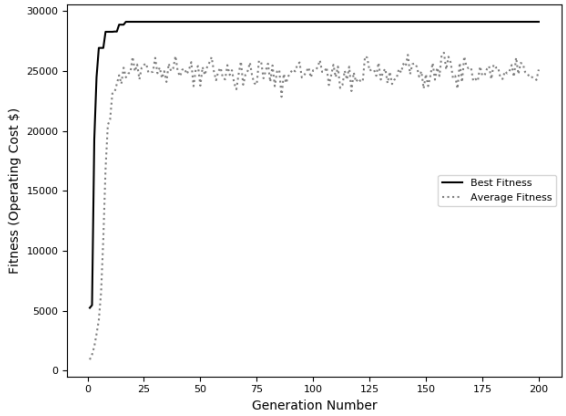
shows the best and average fitnesses and operating cost, averaged over all runs. It is able to find the best solution around generation 110, and there are no improvements after this. Figure 6.5b and 6.5c show the best and average fitness values obtained for two of the single runs. Figure 6.5b shows a steady increase in the operating cost until generation 16, and then it takes another 45 generations to produce a significantly better offspring. The fittest individual is found at just over 100 generations, causing an operating cost of $30,467. The evolution shown in Figure 6.5c finds a solution that is particularly high very early in the evolution. It finds a good individual, as shown by the sharp increase achieved around generation 5 which has an operating cost of $26,934. As the evolution continues, further improvement is found in maximum fitness (to $29,116) and, after generation 17, no improvement is observed. Given the dimensionality of the plant, it is not clear if this is the best that can be achieved. For example, the best individual obtained in the run shown Figure 6.5b that resulted in the operating cost of $31,450, was not the best individual in another run, shown in Figure 6.5c. In the latter, it had an operating cost of $27,537 in comparison to the best individual in this run at $29,116. In the TE Model, this variability is a consequence of using a new seed for the random number generator for each simulation run [18]. This means that two runs are never the same.



(a) Best and average fitness (averaged over all 6 runs)



(b) Best and average fitness for Run I



(c) Best and average fitness for Run II

Figure 6.5: GA results: maximum (best) and average fitness obtained for evolving DoS attacks

As a result, a chromosome with the same genetic material may yield different fitness scores in different runs because of the variability in the operating costs, caused by the randomness of the plant. It is possible that this is the optimal point for this particular run. It is also possible that the GA converges prematurely to a local optimum. This is a common problem in GAs, in which individuals with better finesses are favoured over those that are less fit. This favouritism helps to produce individuals that are more fit than their early ancestors, but it also causes the diversity of the population to decrease. When the global optimum is not known (which is the case for open problems, as in our case), the only reference points we have are the cost of the single attacks. To consider the variability caused by the randomness of the plant, different runs with same genetic operators and the same initial populations were tested. This resulted in individuals with the same chromosome having different fitness scores.

6.1.4.1 Discussion

The results obtained show that the GA is able to search the space efficiently, and give some estimate of potential damage attacks could cause on the plant. However, a more comprehensive set of experiments are required to test how much the randomness of the plant influences these results and whether it is possible to select attacks that have higher expected fitness in such a stochastic environment. It would be desirable to carry out multiple experiments using different parameters on the same plant (using the same seed) to control convergence, and to determine if it is possible to improve the best known solution. Maintaining strong diversity in the population can be beneficial to reduce the risk of getting stuck in the current local optima, and encourage the evolution to converge to better solutions. Strategies that can enhance the genetic diversity include: increasing mutation rate; increasing the population size; controlling the resemblance of the offspring to parents (for example, restricting mating individuals whose genetic material is too similar - incest prevention); injecting new randomly generated individuals into the gene pool (known as random immigrants); using island modelling; using diversity guided algorithms; and using selection strategies that promote diversity (e.g. fitness uniform selection scheme)

[294].

Unfortunately, the requirements for additional computational resources prevented an exhaustive comparison.

The results answer the first part of our Research Question 2: we are able to search the space and find attacks that are better than those single attacks. Before moving on to optimising our attacks, we carried out some experiments to analyse how well the GA is able to cope with a larger space of attacks. In this section, we asked the GA to evolve DoS attacks, in the following section, we use the GA to include IntegrityMin and IntegrityMax attacks in addition to DoS.

6.1.5 Evolving Multiple Attacks using Genetic Algorithm

GA was used to search the attack space containing both DoS and Integrity attacks. For this task, genes are defined using a range of numbers (0=Do not, 1=DoS, 2=IntegrityMax and 3=IntegrityMin). The GA was configured using the parameters in Table 6.3. The chromosome size was 25, equal to the number of sensors and actuators to attack. As before, the initial population of attacks were generated randomly.

Parameters	Value
Chromosome size	25
Types of genes	4
Chromosome encoding	Integers
Description of genes	Do not Attack, DoS, IntegrityMin, IntegrityMax
Number of generations	400
Parent population size	200
Offspring population size	200
Crossover probability	0.8
Mutation probability	0.2
Probability of mutating a gene in a chromosome	0.05
Crossover operator	Two-point crossover
Mutation operator	Uniform
Selection operator	Tournament
Parallel evaluations	25
Computation time	$\approx$ 20 hrs

Table 6.3: Evolutionary operators and parameters for evolving multiple types of attacks using GA

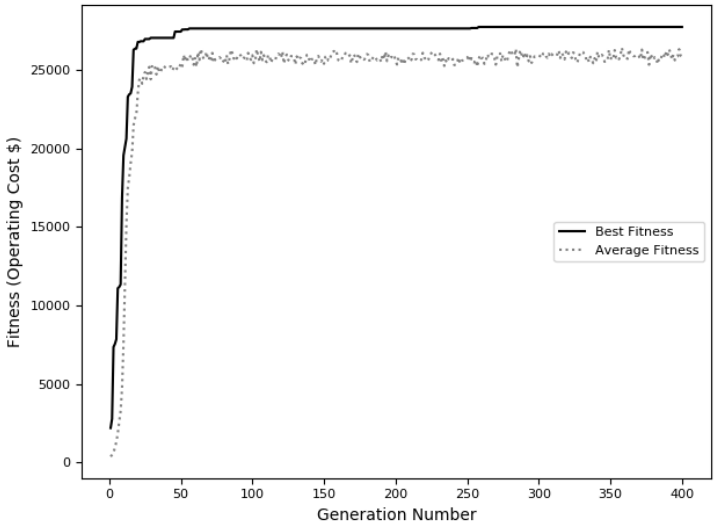
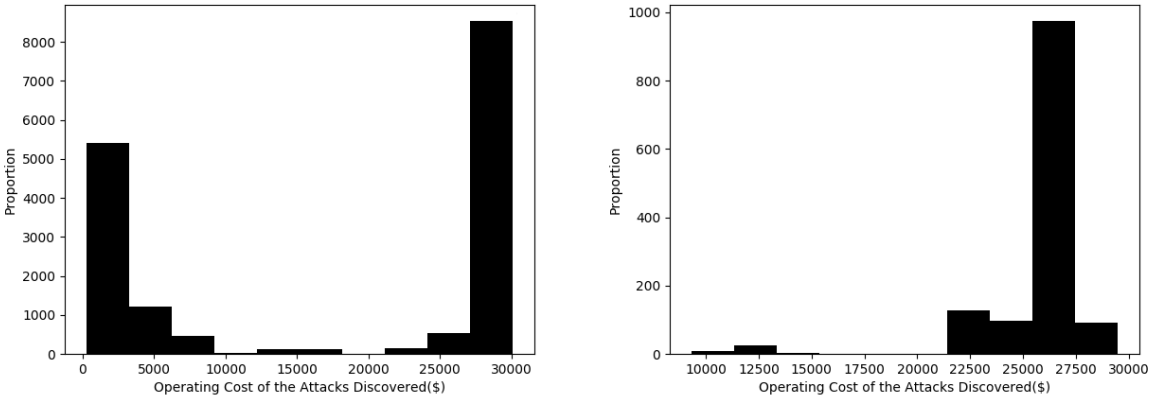


Figure 6.6: GA showing the maximum (best) and average fitness rates for multiple attacks (averaged over all runs)



(a) All Attacks **(b)** Attacks avoided Shutting-down of Plant

Figure 6.7: Generation of multiple types of attacks using GA

Figure 6.6 shows the best and average fitness value for each generation, averaged over 6 runs. The results were comparable to the DoS-only attack set. The GA was able to find attacks that increased the operating cost significantly more than single attacks (to over $30,000); the best attack it found is listed in Table 6.4. On average, the GA was able produce just over 1300 attacks that cause damage without turning the plant off. The histograms in Figure 6.7 show the attacks found in one of the runs: similar results were obtained for other runs. The number of attacks found is significantly higher than the number of DoS-only attacks found previously. However, although this yielded a higher number of attacks, it also confirmed why optimisation is required.

Chromosome	0, 0, 2, 2, 1, 3, 0, 0, 2, 0, 0, 0, 0, 0, 0, 3, 0, 3, 2, 3, 1, 1, 1, 0, 1
Description	$XMEAS1_{NOATTACK}$, $XMEAS2_{NOATTACK}$, $XMEAS3_{IntegrityMin}$, $XMEAS3_{NOATTACK}$, $XMEAS4_{IntegrityMin}$, $XMEAS4_{NOATTACK}$, $XMEAS5_{DoS}$, $XMEAS5_{NOATTACK}$, $XMEAS7_{IntegrityMax}$, $XMEAS8_{NOATTACK}$, $XMEAS9_{NOATTACK}$, $XMEAS10_{IntegrityMin}$, $XMEAS10_{NOATTACK}$, $XMEAS11_{NOATTACK}$, $XMEAS12_{NOATTACK}$, $XMEAS14_{NOATTACK}$, $XMEAS15_{NOATTACK}$, $XMEAS17_{NOATTACK}$, $XMEAS31_{NOATTACK}$, $XMEAS40_{IntegrityMax}$, $XMV1_{NOATTACK}$, $XMV2_{IntegrityMax}$, $XMV3_{IntegrityMin}$, $XMV3_{NOATTACK}$, $XMV4_{IntegrityMax}$, $XMV6_{DoS}$, $XMV6_{NOATTACK}$, $XMV7_{DoS}$, $XMV7_{NOATTACK}$, $XMV8_{DoS}$, $XMV8_{NOATTACK}$, $XMV10_{NOATTACK}$, $XMV11_{DoS}$, $XMV11_{NOATTACK}$
Cost ($)	30107

Table 6.4: GA generated attack strategy utilising multiple types of attacks

Figure 6.6 shows the evolution was slower in comparison to DoS-only attacks, and this is as expected because the search space is larger (4^{25}) and it took more generations to converge.

Although the GA is able to find solutions fairly quickly, and gradually enhance

the average and maximum fitness until around generation 75, the results from all 6 runs appeared to suffer from premature convergence, so it is possible they were not reaching the global optimum and were, instead getting trapped in local optima. Increasing the mutation rate and population size [295] improved the premature convergence only slightly; it is possible that the development of a more advanced GA would help with the problem. This could be done using genetic diversity preserving techniques [294]; however, it is also possible to encourage this by formulating the problem as an multiobjective optimisation, and using diversity as one of the objectives [236] to slow the convergence rate and widen the area of search. This is left as future work.

The results obtained appear to show ability to generate better attacks. The GA is able to produce a rich set of attacks using a small number of primitives. However, this study also illustrates why optimisation is required. In this current setting, it is not possible to determine if all these variables were required to cause this damage. These attacks are not least effort and, thus, there are possible attacks that could cause the same damage by attacking only a subset of these variables. One of best solutions, in Table 6.4 necessitates attacking a combination of 13 sensors and actuators. It is thus important to search for the attacks that are least effort to identify the most vulnerable XMEAS and XMV variables.

A possible option to improve this GA is to consider the cost of the attack by giving more weight to least effort attacks and, in this way punish those involving high numbers of sensors and actuators. This is a common approach in single-objective evolutionary algorithms: to combine multiple objectives into a single function using methods such as utility theory and weighted sum methods [296]. However, the disadvantage of this option is that designing an aggregating function such as a weighted sum (e.g. $Weight1 \times$ObjectiveA + $Weight2 \times$ObjectiveB) is often non trivial. We decided to solve this problem using evolutionary multiobjective optimisation.

6.2 Searching Attacks using Evolutionary Multiobjective Optimisation

To investigate the second part of our Research Question 2: Can we optimise these attacks in terms of effort (number of sensors and actuators attacked) required and damage caused (in terms of safety and economic damage) to determine the most vulnerable combinations?, we decided to formulate the search as an evolutionary multiobjective optimisation (EMO) problem. As we discussed in Section 4.3, in evolutionary multiobjective optimisation each solution is a trade-off between competing objectives, and the evolution tries to find multiple optimal trade-offs, where the measure of optimality is often based on Pareto dominance. Figure 6.8, presents an optimal set containing non-dominated solutions in the decision space with the Pareto front set shown in red. Obtaining this trade-off, and filtering out the dominated solutions, can be carried out in a more efficient way using evolutionary multiobjective optimisation as they evolve a population of solutions simultaneously at each iteration [297].

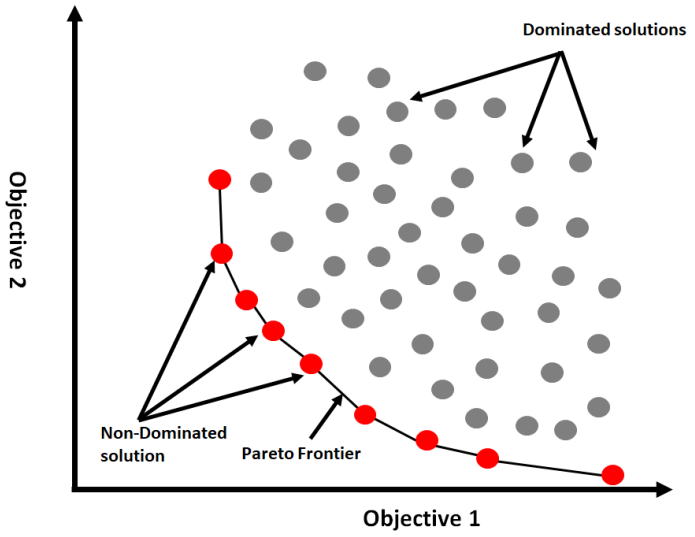


Figure 6.8: Example of Pareto front of a two objective (min-min problem) and the set of solutions

The selection methods of two common and reputable multiobjective evolutionary algorithms, NSGA-II and SPEA2, were used as selection operators in our EMO

to achieve multiobjective optimisation. The genetic operators for varying the population (crossover, mutation or reproduction) that were used to modify the genes of the chromosomes and their parameters were selected based on preliminary tests done to tune the parameters. After testing we found our operators increased the performance and exploration ability of the EMO better than those suggested for NSGA-II and SPEA2 algorithms (Section 4.3.2).

We compare the performance and determine how effective and efficient our EMOs are at finding the optimal attacks against the sensors and actuators. In this section we analyse and show the performance of the EMOs for two scenarios: i) search for the fastest attacks that cause the plant to shut down using the least effort; ii) search for the attacks that cause the most economic damage while keeping the plant operating until the end of the process (chemical production) using the least effort.

In the following subsections, we first explain the performance metrics used for comparing the performance of the selection operators, NSGA-II and SPEA2. Next, we explain the generation of EMOs for the two scenarios and report the results.

Evolutionary multiobjective optimisation is different from the traditional single objective GA used in the previous section, as there is more than one objective. Measuring the performance of these multiple and conflicting objectives can be difficult [298], especially if the problem is large and complex. According to Zitzeler [233] a good EMO should have three goals: i) to minimise the distance of the non-dominated solution set to the true Pareto-optimal front; ii) to find good, in most cases uniform distribution of solutions; iii) to maximise the extent of the resulting non-dominated front. A wide variety of metric indicators have been proposed to compare the performance of evolutionary optimisation algorithms. Often the goal of these metrics is to determine [299]: the distance between the true Pareto front (if it is known) and the predicted Pareto front; the convergence to true Pareto front; the number of elements (cardinality) found in the Pareto front set; how well the true Pareto front is discovered by the obtained Pareto front set; how well they approximate the true Pareto front; and the volume covered by the solutions in the obtained

Pareto front. A comprehensive list of these metrics can be found in [299]. Given that the true Pareto fronts of the problem space we are trying to solve are unknown, the following two metrics will be used to compare the performance of the EMO algorithms:

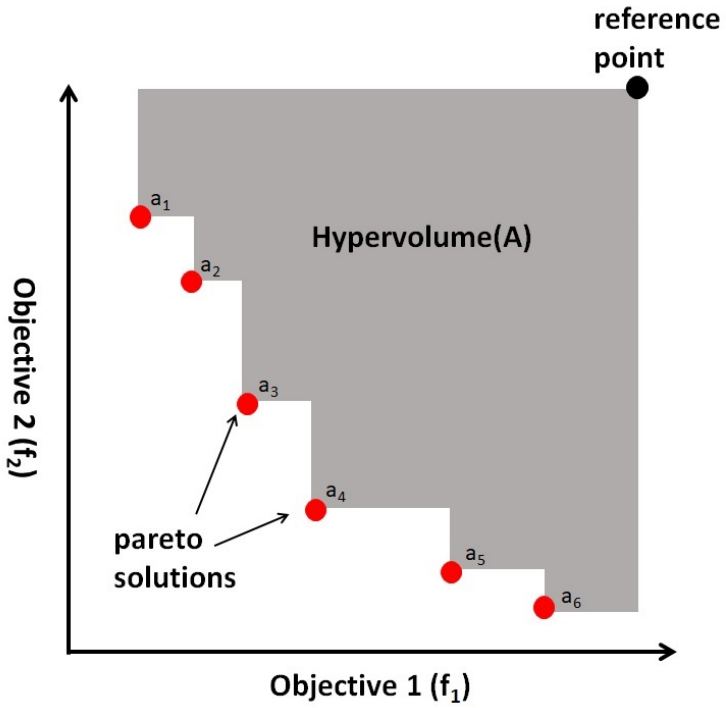


Figure 6.9: Hypervolume indicator in the 2-objective case: Pareto front solutions (red dots) and hypervolume (grey area) [300]

- The *Hypervolume Indicator[246]*, also known as the *Lebesgue measure* [301] or *S metric* has been widely used for comparing the performance of EMO algorithms. It measures the size of volume between the estimated Pareto front and a reference point. The reference point is defined as the worst possible point for the space (sometimes called anti-optimal point). Figure 6.9 shows the hypervolume (grey area) for two objectives, with solutions in which non-dominated Pareto individuals are in red and the other solutions are in the grey area. The choice of the reference can influence the comparison of the solutions, and thus it needs to be carefully selected. The selection of this reference point is an open problem but a common suggestion is to take the worst known values for each of the objectives and shift it slightly towards some unattain-

able values to ensure it is placed in a way that will be dominated by everything else. The same reference points are used to compare the performance of the evolutionary multiobjective optimisation. The goal is to maximise the hypervolume, so a Pareto front with a larger hypervolume is considered to indicate a better set of trade-offs between the objectives. To calculate the hypervolume, we used the Python library implemented by the researchers at TU Dortmund [300], based on the proposed computation algorithm in [302], to track the hypervolume at each generation.

- *The number of solutions* is used to determine the number of attacks found, and the cardinality and the quality of the non-dominated solutions in the Pareto front. Although the optimisation returns the obtained Pareto front, it is also important to determine the number of feasible attacks the optimisation finds as this information will help to identify the most vulnerable sensor and actuator combinations.

6.2.1 Evolutionary Multiobjective Optimisation Approach

Before explaining the whole process of our evolutionary multiobjective optimisation, first, a brief explanation about evolutionary strategies. The (μ,λ) strategy and the (μ+λ) strategy [303] are two classic evolutionary strategies that are often used with optimisation algorithms to determine which individuals will be considered to be competitors in the selection of the next generation. The symbol μ denotes the size of the parent population, and the symbol λ represents the size of the offspring population. These strategies were proposed as part of the evolutionary strategy field in the 1960s by the German researchers for continuous black-box optimisation problems. Since then, they have been widely used in evolutionary computation. In the case of (μ+λ), the next generation population is selected from among the μ parent individuals and the λ offspring. In the case of comma selection, (μ,λ), the selection takes place only among the λ offspring; their parents are disregarded.

When tuning the parameters for our algorithms, the results obtained from our test experiments showed that by using the (μ,λ) strategy we were able to delay

Algorithm 6: Evolutionary multiobjective optimisation algorithm for generating attacks

Input : μ = parents to select for next generation,
λ = offspring to produce at each generation,
$mutpb$ = mutation rate, $cxpb$ = crossover rate,
$ngen$ = number of generation,
$selOp$ = selection operators [NSGA-II, SPEA2]

Output: Pareto Front

```
1  Function Main(μ, λ, mutp, cxpb):
2      ParetoFront = []
3      pop = generate initial population randomly
4      pop=evaluate(pop)
5      gen=1
6      while gen ≤ ngens do
7          offspring = vary(pop,λ, cxpb, mutpb)
8          offspring = evaluateOnTEModel(offspring)
9          if selOp is NSGA-II then
10             pop = selectNSGA-II(offspring, μ)
11         else if selOp is SPEA2 then
12             pop = selectSPEA2(offspring, μ)
13         ParetoFront.update(pop)
14         gen = gen + 1
15     return ParetoFront
16
17 Function vary(pop, λ, mutp, cxpb):
18     offspring=[]
19     for i = 0 to λ do
20         random = randomGenerator(0, 1)
21         if random < cxpb then
22             parent1, parent2 = randomlySelectTwoParents(pop)
23             child1, child2 = crossover(parent1, parent2)
24             offspring.add(child1)
25         else if random < cxpb+mutp then
26             child = randomlySelectParent(pop)
27             child = mutation(child)
28             offspring.add(child)
29         else
30             parent = selectParentRandomly(pop)
31             offspring.add( parent)
32     return offspring
```

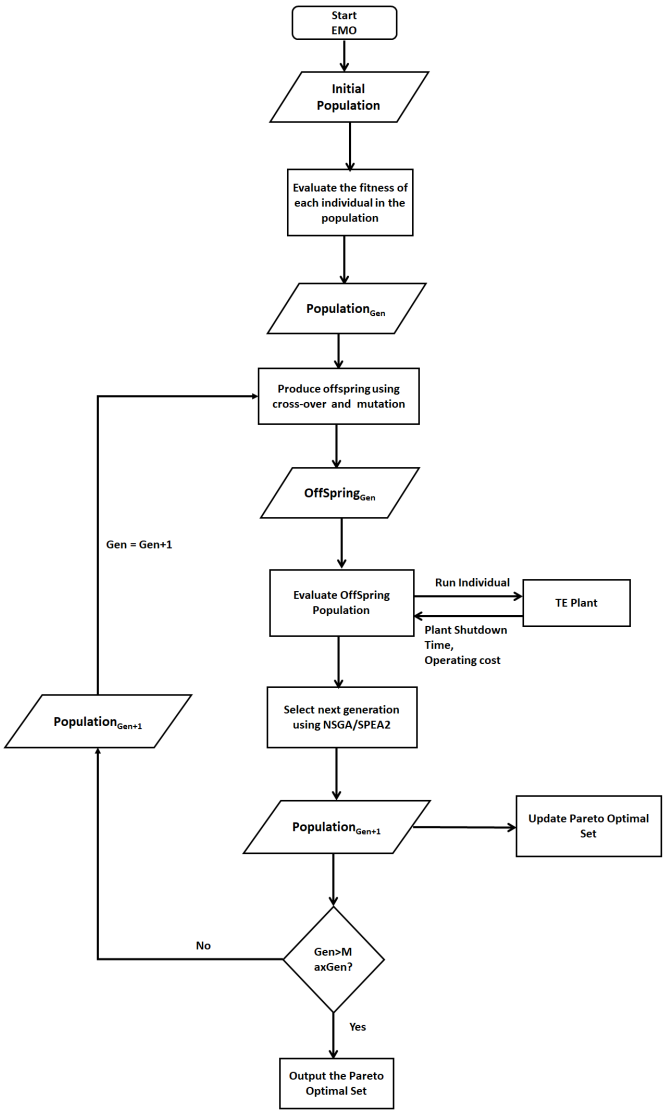


Figure 6.10: Flow diagram of the EMO-based approach designed to generate attacks

the convergence and increase the exploration ability of the search better than for $(\mu+\lambda)$; the results showed that the (μ,λ) strategy converged to a better Pareto front. Therefore, in the following set of studies we used the (μ,λ) strategy. However, a more cautious analysis would require a comprehensive study of both strategies with different sizes of μ and λ.

The attacks are generated and evolved as described in Algorithm 6: the flow diagram in Figure 6.10 gives an overview of structural components of the EMO for generating attacks. In the following section, we discuss the details of our approach.

Chromosome Encoding. Each individual in the population is represented as a

chromosome that consists of 25 genes corresponding to the total number of sensors (XMEAS) and actuators (XMV). The position of the gene denoting a sensor or an actuator and the value of the gene refers to the attack type and parameter.

Initialisation. The initial population of size μ parent chromosomes are generated randomly. The fitness of each chromosome in the initial population is evaluated by converting the chromosome to an attack script written in MATLAB, and the script is executed on the simulated TE model to determine the fitness of the chromosome (i.e. shut down time or the operating cost of the plant obtained from the MATLAB). The operating cost of the plant and plant operating time are written to a file, from which the EMO can collect the fitness of the chromosomes. A pool of MATLAB jobs was assigned to evaluate the fitness of the chromosomes.

Evolutionary Loop (Iterations). After establishing the fitness of the chromosomes in the initial population, the evolutionary loop generates the next generation of chromosomes. First, the genetic variation takes μ parents, and generates λ offspring by applying the variation operators (crossover, mutation or reproduction). The steps involved in this operation are: at each iteration beginning from 0 to λ, randomly chosen chromosomes are subject to one of the three operations: two-point crossover (with a probability of *cxpb*), uniform mutation (with a probability of *mutpb*), or reproduction. If crossover is selected, select two parent chromosomes randomly in the population and apply the crossover operator. If mutation is selected, select a parent chromosome randomly, and apply the mutation operator. In the case of reproduction, one parent chromosome is randomly selected and copied into the offspring population. These steps produce λ offspring. Next, the selection operator (either NSGA-II or SPEA2) is used to select the μ chromosomes from λ size population. The details of the selection method of both algorithms were discussed in Chapter 4. NSGA-II uses non-dominance sorting and crowding distance procedures to rank and select individuals based on where the individual is located. SPEA2 selects individuals based on the number of dominated and non-dominated count as well as their density information. Next, the Pareto front set is updated with the non-dominated chromosomes, and the elements of this set is used to calculate

the hypervolume in each iteration.

Termination Condition. The evolution loop runs for a defined number of generations, and at the end of the evolution, the algorithm returns the set of Pareto front containing all the non-dominated solutions that were found during the evolution.

6.2.2 Using Evolutionary Multiobjective Optimisation for Shutdown Attacks

To evolve attacks that cause the fastest shut down of the plant using the least effort, the multiobjective evolution algorithms were configured using the parameters in Table 6.5. For this evolution, two objectives were defined - f1: minimise the shutdown time (shut down time - attack start time); and f2: minimise the effort required to bring the plant down (total number of sensors and actuators attacked).

Parameters	Value
Chromosome size	25
Types of genes	4
Chromosome encoding	Integers
Description of genes	Do not Attack, DoS, IntegrityMin, IntegrityMax
Number of generations	400
Parent population size (μ)	100
Offspring population size (λ)	200
Selection strategy	(μ, λ)
Crossover probability	0.85
Mutation probability	0.1
Probability of mutating a gene in a chromosome	0.05
Crossover operator	Two-point crossover
Mutation operator	Uniform
Selection operator	NSGA-II, SPEA2
Optimisation objectives	Minimise f1, Minimise f2 f1=shutdowntime, f2=effort
Parallel evaluations	50
Computation time	$\approx$ 27 hrs

Table 6.5: Evolutionary operators and parameters for generating shutdown attacks using EMO

Equal weights were given to each objective because both objectives are equally important. For each evolution, the attack start time and the length of the attack were

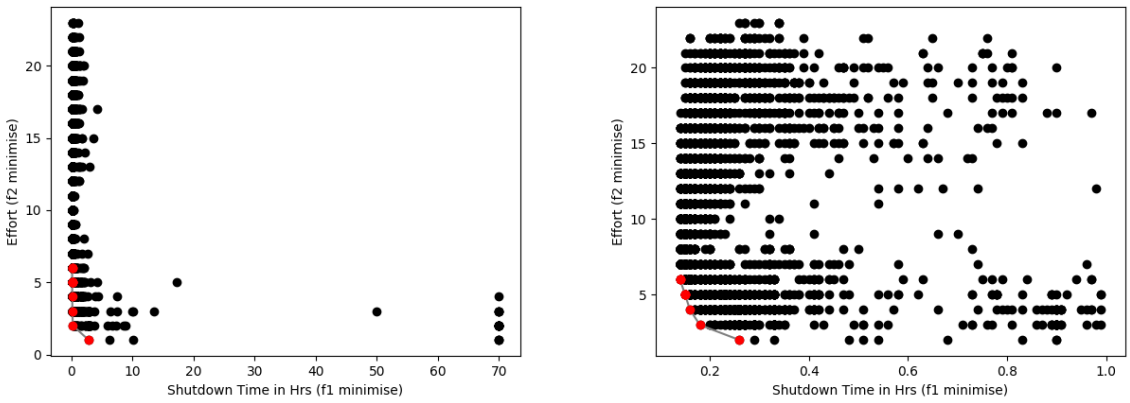
held constant to ensure the conditions were the same for all individuals. Fluctuations in the state of the plant can influence the behaviour of the attack, and thus it was important to keep the conditions the same throughout the evolution. Attacks were started at the same hour, and lasted until the end of the production time. Effort was defined as the total number of sensors and actuators attacked. As shown earlier, in Table 5.3, the results obtained from the randomly generated single attacks showed the fastest shut-down took place using (i) the IntegrityMin attack on XMEAS 4 (A and C Feed flow), causing the plant to shut down in 0.52 hours (31.2 minutes) and (ii) the reactor temperature, with a shut down time of 0.59 hours (35.4 minutes). This is used as baseline against which to compare the performance of the optimisation.

Performance Measure	NSGA-II	SPEA2
Unique attacks	5915	6315
Unique attacks $\leq$ 1hr	5707	5520
Cardinality of Pareto front	15	16
Hypervolume	0.8770	0.8860

Table 6.6: Results for shutdown attacks (averaged over all runs)

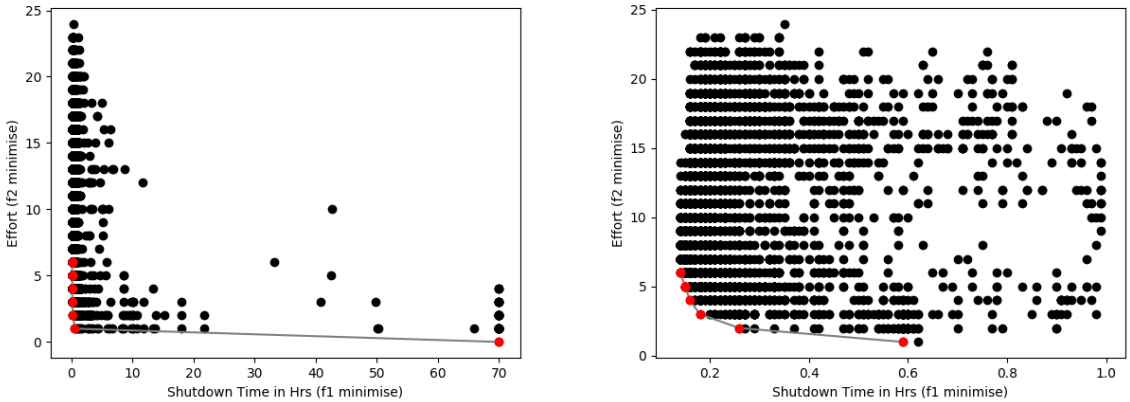
The results were collected over 10 runs for NSGA-II and SPEA2 algorithms. For each of the 10 runs of evolution, a new seed was used to produce a different initial random population to evolve. To compare the results of the optimisation, the same seeds were used for NSGA-II and SPEA2 to ensure both algorithms started with the same initial population. The results obtained were averaged over all runs, and are shown in Table 6.6. After running the evolution for the defined number of generations, the multiobjective optimisation returns the latest Pareto optimal front. Figure 6.11 and Figure 6.12 show the solution space obtained from a single run of the optimisation using NSGA-II and SPEA2.

Both algorithms started from the same randomly generated initial population. The elements of the Pareto front set returned at the end of the final generation are plotted in red dots, and their details are shown in Table 6.7 and Table 6.8. Both optimisation algorithms were able to find almost the same element set in the Pareto front set. SPEA2 outperformed NSGA-II in that it found an extra element in the



(a) All Attacks **(b)** Attacks less than 1hr

Figure 6.11: Shutdown attacks generated using NSGA-II (2-Objective optimisation: minimise shutdown time (f1) versus minimise effort (f2))



(a) All Attacks **(b)** Attacks less than 1hr

Figure 6.12: Shutdown attacks generated using SPEA2 (2-Objective optimisation: minimise shutdown time (f1) versus minimise effort (f2))

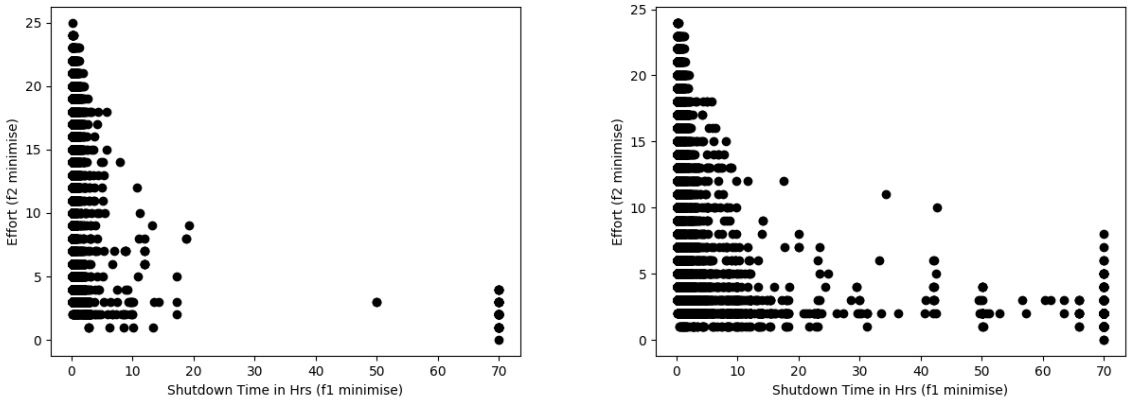
Pareto front set, a total of 13 elements, and found a better solution when effort is 1. For a single, 1-effort attack, the best option founded by NSGA-II was to attack the $\text{XMEAS8}_{\text{IntegrityMin}}$ which takes 2.85 hours and is not the best option. SPEA2 discovered a better attack: an IntegrityMin attack on XMEAS4 will result in faster shut-down of the plant, within 0.59 hours (35.4 minutes).

Attack Strategy	Shut-Down (hr)	Effort
$XMEAS4_{IntegrityMin}$,$XMEAS7_{IntegrityMin}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$, $XMEAS17_{IntegrityMin}$, $XMEAS31_{IntegrityMin}$	0.14	6.0
$XMEAS4_{IntegrityMin}$,$XMEAS7_{IntegrityMax}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$, $XMEAS17_{IntegrityMin}$,$XMEAS31_{IntegrityMin}$	0.14	6.0
$XMEAS4_{IntegrityMin}$,$XMEAS7_{DoS}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$, $XMEAS17_{IntegrityMin}$, $XMEAS31_{IntegrityMin}$	0.14	6.0
$XMEAS4_{IntegrityMin}$,$XMEAS8_{IntegrityMin}$, $XMEAS11_{IntegrityMin}$,$XMEAS17_{IntegrityMin}$, $XMEAS31_{IntegrityMin}$	0.15	5.0
$XMEAS4_{IntegrityMin}$,$XMEAS7_{IntegrityMax}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$, $XMEAS17_{IntegrityMin}$	0.15	5.0
$XMEAS4_{IntegrityMin}$,$XMEAS7_{DoS}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$, $XMEAS17_{IntegrityMin}$	0.15	5.0
$XMEAS4_{IntegrityMin}$,$XMEAS8_{IntegrityMin}$, $XMEAS11_{IntegrityMin}$,$XMEAS17_{IntegrityMin}$, $XMV6_{IntegrityMax}$	0.15	5.0
$XMEAS4_{IntegrityMin}$,$XMEAS8_{IntegrityMin}$, $XMEAS11_{IntegrityMin}$,$XMEAS17_{IntegrityMin}$	0.16	4.0
$XMEAS4_{IntegrityMin}$,$XMEAS8_{IntegrityMin}$, $XMEAS11_{IntegrityMin}$	0.18	3.0
$XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$	0.26	2.0
$XMEAS8_{IntegrityMin}$	2.85	1.0

Table 6.7: Example of a Pareto front obtained for shutdown attacks using NSGA-II

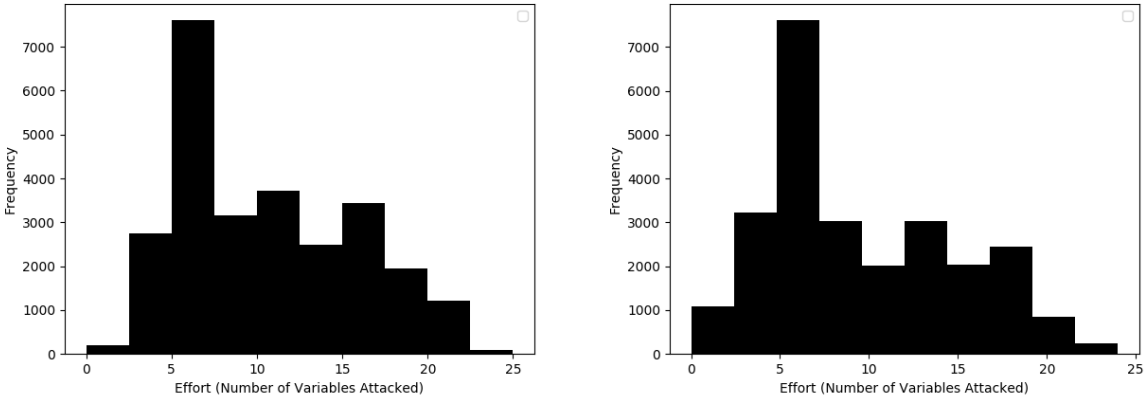
Attack Strategy	Shut-Down Times (hr)	Effort
$XMEAS4_{IntegrityMin}$,$XMEAS7_{DoS}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$, $XMEAS17_{IntegrityMin}$,$XMEAS31_{IntegrityMin}$	0.14	6.00
$XMEAS4_{IntegrityMin}$,$XMEAS7_{IntegrityMax}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$, $XMEAS17_{IntegrityMin}$,$XMEAS31_{IntegrityMin}$	0.14	6.00
$XMEAS4_{IntegrityMin}$,$XMEAS7_{IntegrityMin}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$, $XMEAS17_{IntegrityMin}$,$XMEAS31_{IntegrityMin}$	0.14	6.00
$XMEAS4_{IntegrityMin}$,$XMEAS8_{IntegrityMin}$, $XMEAS11_{IntegrityMin}$,$XMEAS17_{IntegrityMin}$, $XMEAS31_{IntegrityMin}$	0.15	5.0
$XMEAS4_{IntegrityMin}$,$XMEAS7_{DoS}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$, $XMEAS17_{IntegrityMin}$	0.15	5.0
$XMEAS4_{IntegrityMin}$,$XMEAS7_{IntegrityMax}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$, $XMEAS17_{IntegrityMin}$	0.15	5.0
$XMEAS4_{IntegrityMin}$,$XMEAS7_{IntegrityMin}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$, $XMEAS17_{IntegrityMin}$	0.15	5.0
$XMEAS4_{IntegrityMin}$,$XMEAS8_{IntegrityMin}$, $XMEAS11_{IntegrityMin}$,$XMEAS17_{IntegrityMin}$, $XMV6_{IntegrityMax}$	0.15	5.0
$XMEAS4_{IntegrityMin}$,$XMEAS8_{IntegrityMin}$, $XMEAS11_{IntegrityMin}$,$XMEAS17_{IntegrityMin}$	0.16	4.0
$XMEAS4_{IntegrityMin}$,$XMEAS8_{IntegrityMin}$, $XMEAS11_{IntegrityMin}$	0.18	3.0
$XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$	0.26	2.0
$XMEAS4_{IntegrityMin}$	0.59	1.0
Do Not Attack	0.0	0.0

Table 6.8: Example of a Pareto front obtained for shutdown attacks using SPEA2



(a) Shutdown attacks generated using NSGA-II **(b)** Shutdown attacks generated using SPEA2

Figure 6.13: Shutdown attacks generated using NSGA-II and SPEA2 (over all runs)



(a) Distribution of shutdown attacks generated using NSGA-II **(b)** Distribution of shutdown attacks generated using SPEA2

Figure 6.14: Distribution of shutdown attacks generated using NSGA-II and SPEA2 (over all runs)

The Pareto front set also had the case for the *Do not Attack* option. This was also true in other runs: SPEA2 was able produce a more distributed set of solutions, and, on average, a slightly better Pareto front set than the NSGA-II algorithm. This is illustrated in Figure 6.13 and Figure 6.14.

As shown in Table 6.6, SPEA2 produced more attacks on average under 1 hour than NSGA-II, that is 6315 for SPEA2, and 5915 for NSGA-II. The attacks that take down the plant under 1 hour were higher for NSGA-II (5707) compared to SPEA2

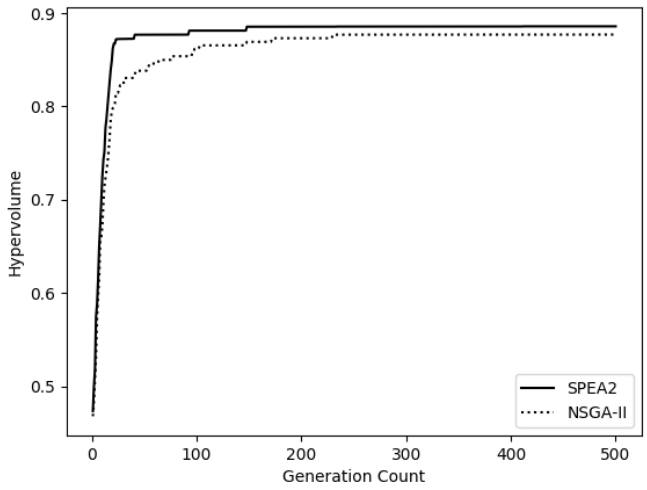


Figure 6.15: Hypervolume results for NSGA-II and SPEA2 for generating shutdown attacks (averaged and normalised over all runs)

(5520). Thus, it is possible that NSGA-II is able to search the space near the optimal set better than SPEA2. Future work will investigate an appropriate metric to study this.

Figure 6.15 shows the performance of the hypervolume indicator for each of the 500 generations, averaged over all runs. At the end of each generation, the hypervolume was computed according to the Pareto front achieved at that generation. For convenience, plotted hypervolume results are normalised to the interval [0,1] according to the best hypervolume value possible, estimated based on the maximum measurement obtained. The reference point used for computing the hypervolume is r =[71, 26]: 71 is the value for objective f1 (shutdown time) and 26 is value for f2 (effort).

The hypervolume results show the speed of the convergence for both SPEA2 and NSGA-II algorithms. As shown in Figure 6.15, SPEA2 converges faster to a better Pareto front whereas NSGA-II requires more time to reach a slightly worse Pareto front set. By generation 148, SPEA2 achieved a value of 0.8855, and from then it converges, improving only slightly to reach 0.8860 by the end of the generation 500, but, overall, outperforming NSGA-II. NSGA-II achieved a hypervolume of 0.8769 around generation 230, and then converges, improving only slightly and

finishing with a hypervolume of 0.8770 by the end of generation 500.

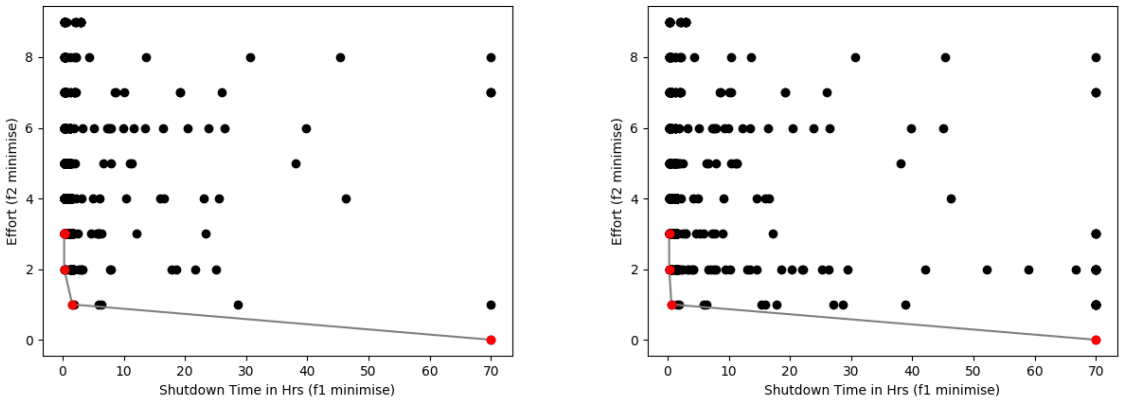
6.2.2.1 Exploring Attack Space for Process Variables and Manipulated Variables Further

The evolutionary multiobjective optimisation approach returned a wide variety of attacks with a range of trade-offs for effort and shutdown time. Most of these attacks involved attacking sensors as we expected, because the single attacks showed that attacks on sensors were more damaging than those on actuators. To determine if this is actually the case, and to determine the individual vulnerability of these components, further experiments were carried out on each set of actuators and sensors separately.

Evolving attacks against manipulated variables (actuators)

Given that the search space for the actuators is much smaller ($4^9 = 262144$), the size of the initial parent population was set to 50 individuals, and number of offspring to produce at each generation was set to 100. The evolution was run for 200 generations using the genetic parameters shown in Table 6.5.

Figure 6.16 illustrates the solution space for NSGA-II and SPEA2. The description of the Pareto front is shown in Table 6.9 and Table 6.10.



(a) Pareto front obtained using NSGA-II **(b)** Pareto front obtained using SPEA2

Figure 6.16: Shutdown attacks generated against XMVs using NSGA-II and SPEA2

The results indicate NSGA-II keeps the evolved populations closer to the non-dominated solutions it finds and, as a result, it was able to find two extra solutions

Attack Strategy	Shut-Down Times (hr)	Effort
$XMV6_{IntegrityMin}$,$XMV10_{IntegrityMin}$, $XMV11_{IntegrityMin}$	0.26	3
$XMV6_{DoS}$,$XMV10_{IntegrityMin}$, $XMV11_{IntegrityMin}$	0.26	3
$XMV4_{IntegrityMax}$,$XMV10_{IntegrityMin}$, $XMV11_{IntegrityMin}$	0.26	3
$XMV10_{IntegrityMax}$,$XMV11_{IntegrityMax}$	0.27	2
$XMV11_{IntegrityMax}$	1.59	1
Do Not Attack	0.0	0

Table 6.9: Description of the Pareto front set for shutdown attacks generated against XMVs using NSGA-II

Attack Strategy	Shut-Down Times (hr)	Effort
$XMV4_{IntegrityMax}$,$XMV10_{IntegrityMin}$, $XMV11_{IntegrityMin}$	0.26	3
$XMV10_{IntegrityMax}$,$XMV11_{IntegrityMax}$	0.27	2
$XMV11_{IntegrityMin}$	0.64	1
Do Not Attack	0.0	0

Table 6.10: Description of the Pareto front set for shutdown attacks generated against XMVs using SPEA2

with effort three. However, during this evolution it found a worse solution for the single effort attack; that is, attacking only one actuator. Given the option to attack one actuator, the best strategy NSGA-II found was an IntegrityMax attack on XMV 11 which will bring down the plant in 1.59hr, whereas SPEA2 found an IntegrrityMin atttack on XMV11, that will result in failure in 0.64 hours. Figures 6.16a and 6.16b shows that SPEA2 is better at searching a more diverse population, generating new solutions in regions NSGA-II fail to explore, and NSGA-II is better at finding solutions near the solution space's best trade-offs.

If the attacker only has access to actuators, the best strategy to bring the plant down is to attack the valve XMV 10 (Reactor Cooling Water Flow) and valve XMV 11 (Condenser Cooling Water Flow). This solution provides the best trade-off between the two objectives. However, the attacker will need to devise the attack carefully, deciding what type of attack to carry out on each actuator. The EMOs found that using $XMV10_{IntegrityMax}$ and $XMV11_{IntegrityMin}$ will shut down the plant in 1.12

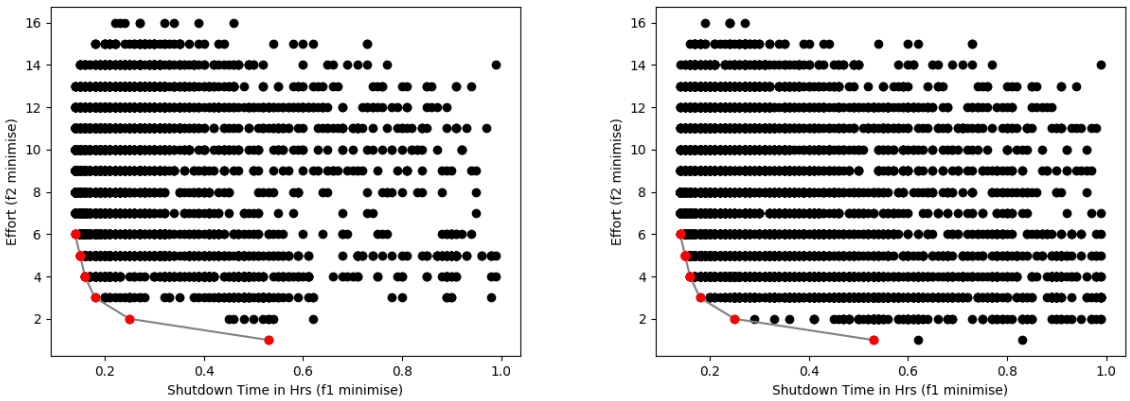
hours, but, using $XMV10_{IntegrityMax}$ with $XMV11_{IntegrityMax}$ will shut down the plant in 0.27 hours (16.2 minutes).

Evolving attacks against process variable measurements (sensors)

We also evolved attacks using only XMEAS sensors to test how these compared to the combined XMEAS and XMV attacks using the genetic parameters shown in Table 6.5.

The results were very similar to the previous results obtained from evolving combined attackers using all XMEAS and XMV variables. The shutdown time was not improved and there were no better attacks to reduce the shut down time of the plant faster; however, the evolution was able to improve the size of the Pareto front set. There were 20 elements in the Pareto set for SPEA2 and 18 elements in the Pareto set for NSGA-II. Figure 6.17 shows the attacks that cause failures under 1 hour for both of the optimisation algorithms. The plot for the Pareto front may not be sufficiently long, but some of the points on the diagram refer to multiple attacks with equal fitness; for example, there were 3 attacks with equal fitness using 6 effort and shutdown time of 0.14 hours; 9 attacks with 5 effort and shut down time of 0.15 hours. This is due to quantisation effects. When converting the shutdown time into hours, we converted to nearest two decimal place, and, in reality these attacks are very slightly different. For our experiments, this time granularity was sufficient, but temporal granularity can be configured according to requirements.

We will not list all the attacks in the Pareto front, but the solutions with effort ≤ 4 obtained from EMO algorithm using the SPEA2 selection operator are listed in Table 6.11. When compared to Table 6.8, there are three more attacks using effort four. For this experiment, the population size was kept the same as for the combined attacks as in the previous experiment but the chromosome size was 16 smaller than the combined size of sensors and actuators (25). This might have improved the quality of the Pareto set for this exercise, resulting in mating more offspring of similar genetic material. One might achieve similar results with the combined attacks if the size of the population is increased, since the size of the population can influence the quality of the Pareto set.



(a) Attacks under 1hr using NSGA-II **(b)** Attacks under 1hr using SPEA2

Figure 6.17: Shutdown attacks generated against XMEASs using NSGA-II and SPEA2

Attack Strategy	Shut-Down (hr)	Effort
$XMEAS4_{IntegrityMin}$,$XMEAS7_{IntegrityMin}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$	0.16	4.0
$XMEAS4_{IntegrityMin}$,$XMEAS7_{IntegrityMin}$, $XMEAS8_{IntegrityMin}$,$XMEAS17_{IntegrityMin}$	0.16	4.0
$XMEAS4_{IntegrityMin}$,$XMEAS7_{DoS}$, $XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$	0.16	4.0
$XMEAS4_{IntegrityMin}$,$XMEAS7_{IntegrityMin}$, $XMEAS8_{IntegrityMin}$,$XMEAS31_{IntegrityMin}$	0.16	4.0
$XMEAS4_{IntegrityMin}$,$XMEAS8_{IntegrityMin}$, $XMEAS11_{IntegrityMin}$	0.18	3.0
$XMEAS8_{IntegrityMin}$,$XMEAS11_{IntegrityMin}$	0.25	2.0
$XMEAS4_{IntegrityMin}$	0.53	1.0
Do Not Attack	0.0	0.0

Table 6.11: Description of some of the elements in the Pareto front for shutdown attacks against XMEASs generated using SPEA-2

6.2.3 Using Evolutionary Multiobjective Optimisation for Economic Damage Attacks

We investigated evolutionary multiobjective optimisation to evolve attacks that cause economic damage using the least effort. The evolutionary parameters used for searching for these attacks are reported in Table 6.12. The attacks discussed in the previous sections started at the same time. To cause monetary loss, the attackers need to run attacks for longer periods whilst avoiding detection. Given the increased

Parameters	Value
Chromosome size	25
Chromosome encoding	Integers
Types of genes	36
Chromosome encoding	Integers
Description of genes	Do not Attack, DoS, IntegrityMin, IntegrityMax, Replay, (with various starting time and duration)
Number of generations	500
Parent population size (μ)	200
Offspring population size (λ)	400
Selection strategy	(μ, λ)
Crossover probability	0.80
Mutation probability	0.15
Probability of mutating a gene	0.08 in a chromosome
Crossover operator	Two-point crossover
Mutation operator	Uniform
Selection operator	NSGA-II, SPEA2
Optimisation objectives	Maximise f1, Minimise f2 f1=operating cost, f2=attack effort
Parallel evaluations	50
Computation time	$\approx$ 66 hrs

Table 6.12: Evolutionary operators and parameters for generating economic damage attacks using EMO

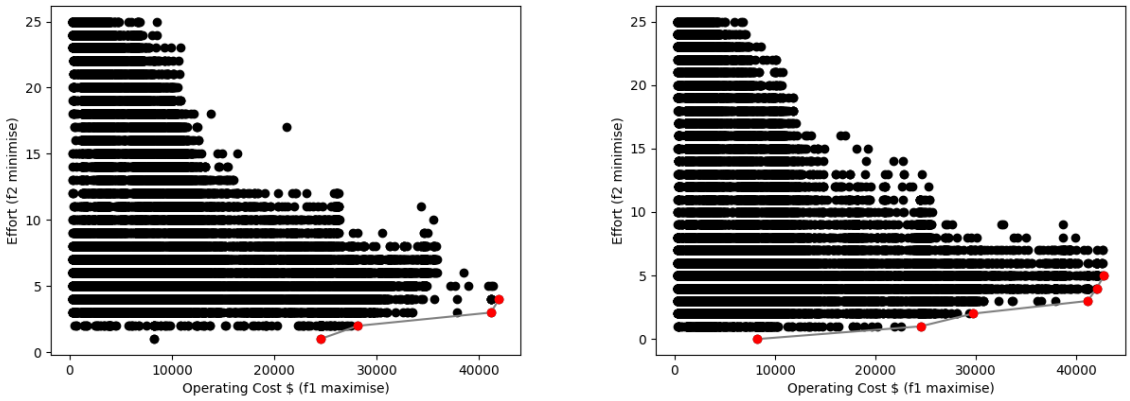
length of the computation time, as shown on Table 6.12, we decided to configure the attacks using only a limited set of attack times: starting time of the attacks (2, 10, 20, 30, 50 hours) and duration of attacks (10, 20, 22, 42, 52, 62, 70 hours).

The length of the attacks was kept long to estimate the maximum potential economic damage inflicted on the plant. In total there were 36 attack strategies (types of genes) for each chromosome, and this is a search space of size 36^{25}. The optimisation is a 2-objective fitness problem: maximise f1 and minimise f2, where f1 is the total operating cost of the plant; and f2 is the effort (total number of sensors and actuators attacked). The results were obtained over 10 sets of runs for each optimisation algorithm. Table 6.13 reports the results obtained.

EMO was able to find attacks that were better than the single attacks reported in Section 5.1, random search in Section 6.1.4 and the single objective GA dis-

Performance Measure	NSGA-II	SPEA2
Unique attacks	44846	44105
Maximum operating cost ($)	41938	42748
Cardinality of Pareto front	5.0	6.0
Hypervolume	0.8202	0.8608

Table 6.13: Results for operating cost attacks (averaged over all runs)



(a) Attacks generated using NSGA-II

(b) Attacks generated using SPEA2

Figure 6.18: Example of economic attacks generated using NSGA-II and SPEA2 (single run)

cussed in Section 6.1.5. The average operating cost of the plant is $8208. Starting from a random population of individuals, multiobjective evolutionary computation went through over 44,000 unique attacks on average, and increased the operating cost to $42,748, a total increase of $34,540 from the average of operating the plant normally.

Figure 6.18 shows the performance of the EMO with the Pareto front set gener-

Attack Strategy	Operating Cost ($)	Effort
$\text{XMEAS7}_{\text{IntegrityMax(2,70)}}$,$\text{XMEAS10}_{\text{IntegrityMin(2,70)}}$, $\text{XMEAS31}_{\text{DoS(10,62)}}$,$\text{XMV3}_{\text{IntegrityMin(2,70)}}$	41938.10	4
$\text{XMEAS7}_{\text{IntegrityMax(2,70)}}$,$\text{XMEAS31}_{\text{DoS(10,62)}}$, $\text{XMV3}_{\text{IntegrityMin(2,70)}}$	41121.75	3
$\text{XMEAS7}_{\text{IntegrityMax(2,70)}}$,$\text{XMV3}_{\text{IntegrityMin(2,70)}}$	28141.06	2
$\text{XMEAS7}_{\text{IntegrityMax(2,70)}}$	24473.12	1

Table 6.14: Description of a Pareto front obtained for economic damage attacks generated using NSGA-II

Attack Strategy	Operating Cost ($)	Effort
$XMEAS7_{IntegrityMax(2,70)}$,$XMEAS10_{IntegrityMin(2,70)}$, $XMEAS31_{DoS(10,62)}$,$XMV3_{IntegrityMin(2,70)}$, $XMV4_{IntegrityMin(10,20)}$	42748.16	5
$XMEAS7_{IntegrityMax(2,70)}$,$XMEAS31_{DoS(10,62)}$, $XMV3_{IntegrityMin(2,70)}$,$XMV4_{IntegrityMin(10,20)}$	42013.06	4
$XMEAS7_{IntegrityMax(2,70)}$,$XMEAS31_{DoS(10,62)}$, $XMV3_{IntegrityMin(2,70)}$	41121.75	3
$XMEAS7_{IntegrityMax(2,70)}$, $XMV4_{Replay(2,70)}$	29675.01	2
$XMEAS7_{IntegrityMax(2,70)}$	24473.12	1
Do Not Attack	8210.42	0

Table 6.15: Description of a Pareto front obtained for economic damage attacks generated using SPEA2

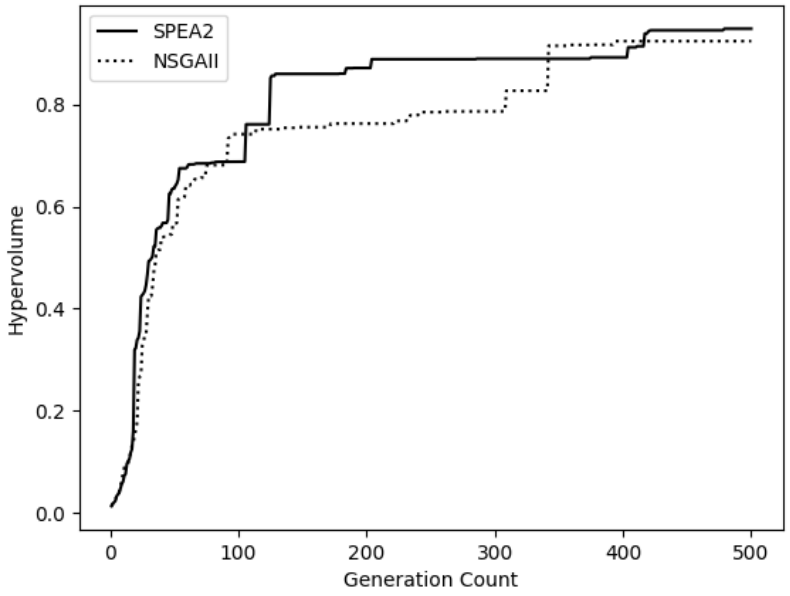
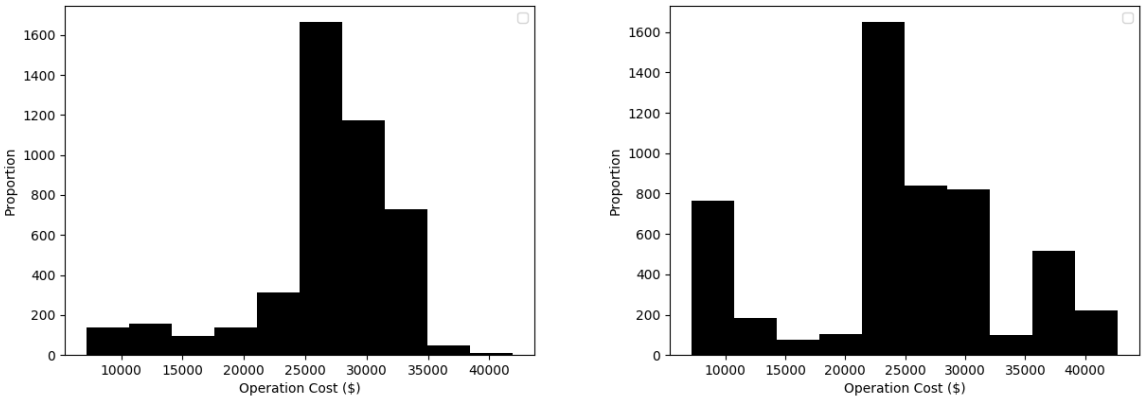


Figure 6.19: Hypervolume results for NSGA-II and SPEA2 (single run)

ated using NSGA-II and SPEA2 selection operators. A description of the Pareto set is given in Table 6.14 and Table 6.15. The values in the tuple show the attack start time and duration, for example $XMV4_{IntegrityMin(10,20)}$ means an integrity attack was carried out on XMV4, starting at hour 10, for a duration of 20 hours. These attacks avoided shutting down the plant, and they were all able to run until the end of the production; in other words the safety limits were not violated during these attacks. At the end of generation 500, the SPEA2 algorithm produced a Pareto set containing

6 elements, and NSGA-II produced a Pareto set containing only 4 elements.



(a) Distributions of attacks generated using NSGA-II **(b)** Distribution of attacks generated using SPEA2

Figure 6.20: Distribution of economic attacks generated using NSGA-II and SPEA2 (single run)

As for previous results NSGA-II required more time to find the Pareto front than SPEA2. The hypervolume of this run is illustrated in Figure 6.19, and this suggests optimisation might have improved if it was run for more generations. Whilst it is not clear that the differences will be major, future work could use checkpoints to continue an evolution beyond the predefined maximum generation criteria, stopping only when search stagnates: that is, that there are no changes in the hypervolume for a number of successive generations.

Figure 6.20 shows the proportion of these attacks that did not violate the safety limits of the plants and kept the plant on, generated throughout the evolution of each of the optimisation algorithms from the same single run. The diversity of the solutions was better for SPEA2, producing 5272 unique attacks (ranging from \$4,2748.16-\$7,150.82) as compared to 4468 unique attacks (\$41,938.1-7,138.48) for NSGA-II.

As these numbers illustrate, the plant was also able to produce some attacks that *reduced* the operating cost to below the mean of the normal runs (\$8208). Moreover, none of previously studied attacks had an harmful effect on the product quality. One possible explanation for this reduction is that it is a consequence of producing less product output. The original TE model does not have a metric

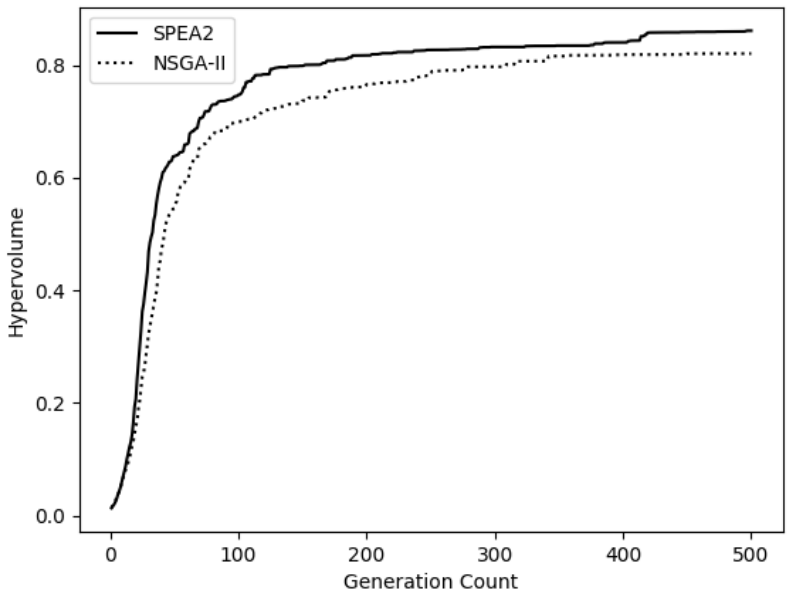


Figure 6.21: Hypervolume results for NSGA-II and SPEA2 for generating economic damage attacks (averaged and normalised over all runs)

for calculating the quantity of the final product of the plant. However, these results suggest that an either there is scope to improve the process, or there is scope to attack this system by forcing it to produce less product.

On average, SPEA2 performed better than NSGA-II both in the quality of the Pareto front and the time it takes to converge. As indicated in Table 6.13, the average over all the runs showed that SPEA2 has a hypervolume average of 0.86, against a NSGA-II hypervolume of 0.82. Figure 6.21 shows the comparison of hypervolume between NSGA-II and SPEA2, normalised and averaged over all 10 runs. These results show that SPEA2 is able to search the space faster and produce a better Pareto front of optimal solutions. Given the scale of the computation required, it was infeasible to re-run these algorithms for a larger number of generations; however, these results suggest both optimisation methods can be used to address this problem.

6.2.4 Discussion

At the beginning of this chapter to investigate Research Question 2 (How should one search the attack space effectively and efficiently and identify the most vulnerable

components of the process?), we broke the question further into two parts:

- Can we search for the attacks that cause the most damage (e.g. operating cost, increase the cost)?
- Can we optimise these attacks in terms of effort (number of sensors and actuators attacked) required and damage caused (damage) to determine the most vulnerable combinations?

The experiments carried out using evolutionary multiobjective optimisation answer the second part of our Research Question. We were able to produce a rich dataset of feasible attacks that can be carried out on the plant to damage its safety or cause economic damage.

The EMO algorithms using either NSGA-II or SPEA2 selection operators were able to produce an optimal set of solutions, but SPEA2 appears to be slightly faster and results in a better quality Pareto front set and spread among the solutions. However, how well the evolved attacks represent the full extent of the trade-offs between the objectives, and how close the non-dominated solution approximation set is to the true Pareto Front can be hard to measure when the true Pareto front is not known. Thus, the result obtained from 500 randomly generated attacks carried out on each sensor and actuator were used as a comparator for these results. By calculating the hypervolume of the Pareto front obtained at each generation, we were able to monitor convergence. Overall, the attacks evolved for shutting down the plant took less time to converge, and attacks evolved to increase the operating costs required more time to converge as the attack space was larger. For practical reasons, we could only run these algorithms for 500 generations. Although this was in general a sensible number for most of the runs, there were a few cases in which the optimisation required more time to converge. So, using the generation number as a termination criterion may not always be appropriate, and population stagnation might be a better stopping criterion.

Premature convergence is a general problem of evolutionary computation and it is important to maintain the diversity of the population to reduce this risk. Increasing the mutation rate, population size and using the (μ,λ) strategy reduced the

risk, but applying more advanced and dynamic techniques might help to find better solutions.

Vulnerable Combinations	SDT (min)
XMEAS4,XMEAS7,XMEAS8,XMEAS11,XMEAS17,XMEAS31	8.4
XMEAS4,XMEAS8,XMEAS11,XMEAS17,XMV6	9.0
XMEAS4,XMEAS8,XMEAS11,XMEAS31	9.6
XMEAS3,XMEAS4,XMEAS8,XMEAS11	10.2
XMEAS8,XMEAS9,XMEAS11	10.8
XMEAS8,XMEAS11,XMV6	11.4
XMEAS8,XMEAS11,XMV10	12.0
XMEAS8,XMEAS11,XMEAS31	12.0
XMEAS5,XMEAS8,XMEAS11	13.8
XMEAS9,XMV7,XMV11	14.4
XMV7,XMV10,XMV11	14.5
XMEAS4,XMEAS7,XMEAS17	15.0
XMEAS8,XMEAS11	15.0
XMV9, XMV10	16.2
XMV10,XMV11	16.2
XMEAS9,XMV11	16.8

Table 6.16: A selection of vulnerable combinations that could bring the plant down under 17 minutes

The results obtained provide a rich set of attacks that can be analysed and help plant operators identify the most vulnerable combinations of sensors and actuators. Table 6.16 shows a subset of vulnerable combinations of the sensors and actuators that could shut down the plant in under 17 minutes. For example, carrying out a single attack on XMEAS 8 (Reactor Level) was able to shut down the plant over 2.8 hours, and attacking XMEAS 11 (Separator Temperature) avoided shutting down the plant for all types of attacks. The results show that attacking both of these sensors at the same time could shut down the plant in 15 minutes. These results also showed that the plant is less resilient to attacks on the sensors and, if an adversary wants to bring down the plant in a very short period of time such as less than 10 minutes, attacking sensors is more likely to cause this to happen than attacking actuators.

Table 6.17 reports the combinations of sensors and actuators attacked and their influence on the operating cost. Most of these attacks involved carrying out an

Vulnerable Combinations	Operating Cost ($)
XMEAS7, XMEAS10, XMEAS31, XMV3, XMV4	42748.16
XMEAS7, XMEAS17, XMEAS31, XMV3, XMV4	42527.14
XMEAS7, XMEAS31, XMV3, XMV4, XMV7	42085.96
XMEAS7, XMEAS31, XMV3, XMV4	42013.06
XMEAS7, XMEAS15, XMEAS31, XMV3	41141.87
XMEAS7, XMEAS31, XMV3	41121.75
XMEAS7, XMV4, XMV10	37865.92
XMEAS7, XMV3, XMV4	36474.95
XMEAS7, XMEAS15, XMV4	35007
XMEAS7,XMV1,XMV4	33846.47
XMEAS7, XMV4	29777.79
XMEAS7, XMV3	29777.79
XMEAS7, XMEAS31	29675.01
XMV4, XMV10	28152.05
XMV3, XMV4	28139.69
XMEAS9, XMEAS31	27019.83
XMEAS31, XMV10	27018.93
XMEAS5, XMEAS31	26995.75
XMEAS7	24479.21
XMEAS31	19611.87
XMV10	11046.02
XMEAS5	13500.75
XMEAS11	10389.3
XMEAS8	10300.15
XMV4	10097.58
XMV6	10095.01
XMEAS9	9134.93
Do No Attack (Average)	8210.94

Table 6.17: A selection of vulnerable combinations that could increase the operating cost of the plant

IntegrityMax attack on XMEAS 7 (Reactor Pressure), which means sending higher values than expected. When the controller receives these values, it attempts try to reduce the pressure by opening the Purge valve, and a high value for the purge rate increases the operating cost of the plant. Carrying out a single attack on XMEAS 7 increases the operating cost to $24479.21 but, as the results indicate, more damage can be inflicted using the right combinations of sensors and actuators.

The use of evolutionary multiobjective optimisation to generate attacks can help in the design of systems that are more resilient to cyber attacks. One way

to use this knowledge would be to consider the vulnerable combinations when designing control network segments. The *zone and conduit model* is a framework for network segmentation to manage security threats for Industrial Automation and Control Systems, recommended as part of the standard such as ISA/IEC 62443. Zones are defined as a group of logical or physical assets sharing common security requirements, and conduits are the paths of communication between the zones [304]. The use of EMO to identify particularly vulnerable combinations of sensors and actuators that cause the plant to shut down or increase the operating cost of the plant means that these can be aggregated in different zones, building a more secure resilient network.

6.3 Summary

In this chapter, we investigated Research Question 2 by developing a random search, a single objective GA, and evolutionary multiobjective optimisation algorithms. The results shows that evolutionary multiobjective optimisation explores the search space better than random search and the single objective GA, and provided a set of solutions that were better. Starting from a random population, having no knowledge about the system, evolutionary multiobjective optimisation was able to capture the system's behaviour, and identify the non-dominated solutions that correspond to trade-offs between the defined objectives. This work shows that evolutionary multiobjective optimisation can successfully be used to generate attacks that cause the most damage, by identifying the most vulnerable components in a complex system automatically. It provides an effective and efficient search approach for cybersecurity tasks. In the next chapter, we investigate a detection model for the generated attacks.

Chapter 7

Attack Detection using Supervised and Unsupervised Learning

In this chapter, we describe an investigation of Research Question 3 (How should one design a novel intrusion detection system to detect attacks?) using the machine learning and deep learning methods we identified in section 4.4. We used the TE model to generate a large set of attacks to test the performance of the detection methods. We explain the process involved in training, tuning and measuring performance of the supervised and unsupervised learning methods. The performance of the detection mechanisms are measured against four types of attacks: IntegrityMin, IntegrityMax, DoS and replay attacks, which are used to manipulate the process variable measurements sent by the sensors (XMEAS), and manipulated variables received by the actuators (XMV).

7.1 Supervised and Unsupervised Learning Methods

To investigate the effectiveness of supervised learning (classification) and unsupervised learning (anomaly detection) we selected the learning methods illustrated in Table 7.1.

To implement the machine learning algorithms, the widely used machine learning library in Python, scikit-learn [92], was selected; to implement the deep learning algorithms (LSTM, GRU, Autoencoder), Keras [281] was used on top of Tensorflow. Keras is an open source high level neural networks API in Python used to

Supervised Learning	Unsupervised Learning
Single Decision Tree (CART)	One-Class SVM
Bagging	Long Short-Term Memory (LSTM)
AdaBoost	Gated Recurrent Unit (GRU)
Random Forest	Autoencoder Neural Network
Support Vector Machine	

Table 7.1: Selected supervised and unsupervised models for IDS

implement deep neural networks, and can be run over other deep learning libraries such as TensorFlow and Theano.

7.2 Dataset Generation and Description

The TE model generates 41 process variable measurements (XMEAS) and 12 manipulated variables (XMV). We assume that intrusion detection is carried out close to the controller, and that the IDS sees both process variable measurements from the sensors and the manipulated variables sent to the actuators. As illustrated in Figure 7.1, if attacks are carried out on the process variable measurements, the IDS receives these values, as does the controller. If the attacks are carried out against the manipulated variables, neither IDS nor the controller will become aware of this directly, and IDS will use the manipulated variables computed by the controller.

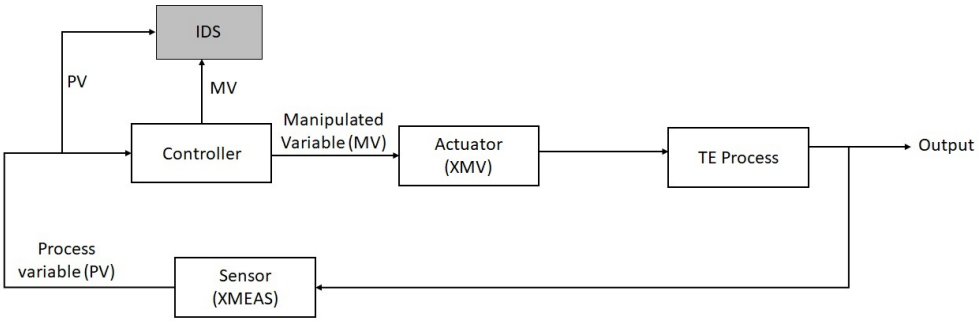


Figure 7.1: Position of the intrusion detection system

The open source fault dataset generated on the TE process model by the Braatz Group [125] uses a sample interval of 3 minutes (20 data points per hour). When dealing with attacks on critical systems, the goal of the detection should be to detect abnormality as rapidly as possible, so that the necessary countermeasures can be put in place before significant damage has occurred. To cater for this, and to test how

Supervised Learning	Samples containing attacks: 60 Samples with normal mode: 12 Attack types: Integrity (Min and Max) attacks carried out on 16 sensors (XMEAS) Duration of attacks: 20 minutes to 3 hrs Validation Data: 20 samples
Unsupervised Learning	Samples with normal mode: 30 Validation Data: 10
Sample size	36000
Sample dimensions	51 41 XMEAS (process variable measurements) 9 XMV (manipulated variables) 1 Attack indicator
Sampling points	500 samples per Hour
Test Data	224 Samples with attacks: 92 Integrity attacks on XMEAS and XMV 56 DoS attacks on XMEAS and XMV 77 Replay attacks on XMEAS and XMV Duration of attacks: 20 minutes to 2 hour

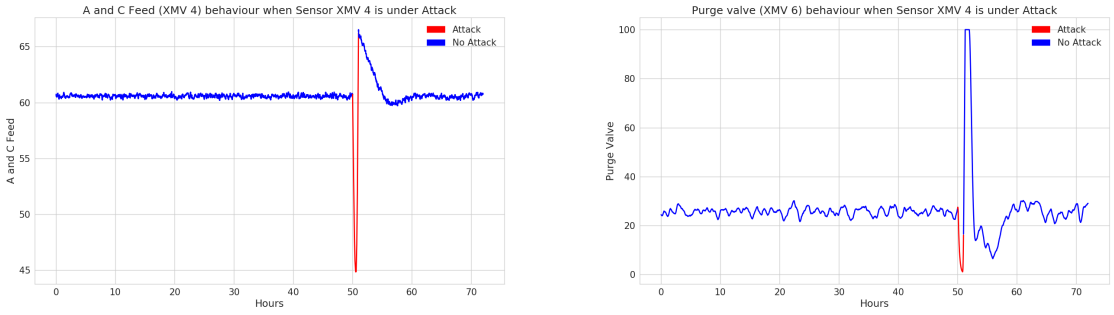
Table 7.2: Characteristics of the IDS dataset

well the learning algorithms cope with a dataset of the size commonly found in real-world deployments, the TE process model was set up to record data at a sampling rate of 500 data points per hour. As a result, each simulated run produces 36000 data points (72 hours x 500) for each of 12 manipulated and 41 process variable measurements. This is just over 30MB of data. All attacks are interval attacks.

Table 7.2 shows the characteristics of the datasets used for designing and testing our methods. Next, we explain the process involved in generating these datasets.

7.2.1 Dataset for Supervised Learning

Supervised learning methods require labelled data for training. We defined the classification as a binary classification problem with two labels: attack and normal. To test the adaptability of training to unseen situations (unknown attacks), the training data for supervised learning were generating using IntegrityMin and IntegrityMax attacks on process variable measurements (i.e. sensors only). To avoid making the task of detection too simple, the simulated attacks were launched against a single sensor during the plant's operation, and for a duration of between 20 minutes and 3 hours to ensure the plant can return to normal operation. The random seed that



(a) Impact on A and C Feed (XMV4)

(b) Impact on Purge Valve (XMV 6)

Figure 7.2: Impact of the attack on A and C Feed (XMV 4)

is used to generate stochastic measurement noise was changed before running each replicate to reflect likely variability in the plant and so avoid overfitting. Using this approach, many attack data cases were simulated.

7.2.1.1 Labelling Data

Supervised learning requires high quality labelled data to ensure that models can learn and classify the phenomena of interest with accuracy. This process can be complicated and costly since the labelling is often done manually by a human expert. The problem is particularly hard for cyber-physical systems, because deciding how to label may not be a straightforward task.

Thus, for example, once attacks are launched, the anomalous behaviour often continues even after the attack has stopped, and it takes a while before the system returns to normal operating conditions. Labelling just the data between the start of the attack and the end of the attack as malicious, and the data elsewhere as normal is simplistic. Figure 7.2a shows an integrity attack that was carried out on XMV 4. The attack starts at hour 50 and runs for a duration of 1 hour but it takes over 10 hours for the plant return to normal operating conditions. Moreover, carrying out an attack on one variable will often have consequences on other variables. Figure 7.2b shows the consequences of this attack on the purge valve. The behaviour between the start of the attack and re-establishment of the normal operation is all the result of the attack; therefore this data should also be labelled as anomalous.

To identify when the plant returns to a normal operating condition after an

attack, a modified Z-score was used. In an ordinary Z-score test, the mean and standard deviation for each feature (XMEAS and XMVs) are used to estimate how many standard deviations an element is from the mean. An observation is considered to be an outlier if the sample is within a given distance of the mean, usually $\pm 3\sigma$. The sample mean and sample standard deviation used in calculating a Z-score can be affected by a small number numbers of outliers. To avoid this, Inglewicz and Hoaglin [305] proposed the modified Z-Score, which uses the sample's median ($\tilde{x}$) and the median of absolute deviation (MAD), instead of the mean and standard deviation:

$$M_i = \frac{0.6745(x_i - \tilde{x})}{\text{MAD}} \tag{7.1}$$

where the constant 0.6745 (the 0.75th quartile of the standard normal distribution) is recommended for larger normal data. MAD is defined as:

$$MAD = median(|Y_i - \tilde{Y}|) \tag{7.2}$$

where Y_i is the median of the sample and $|\tilde{Y}|$ is the absolute value of Y.

Generally, a modified Z-score with an absolute value of ≥ 3.5 is considered to be an indicator that the data is an outlier [305]. A large sample of normal plant operation data were taken to calculate the modified Z-scores for each of 50 features. These scores were used to estimate when the plant returned to normal after the attack; in this case all 50 features returned to their normal range. Two types of labelling were used. Binary classification: *attack* and *not-attack*, and multiple classification, where labels describe what is being attacked: *sensor attack*, *actuator attack* and *not-attack*. However, we decided to focus only on binary classification for two reasons: i) multiple classification was harder to generate reliably; ii) the proposed unsupervised learning methods are binary (anomalous, normal), and therefore comparison would not be straightforward.

7.2.1.2 Forming the Datasets

To train the supervised learning algorithms, two datasets were formed: a training dataset to train the algorithms, and a validation dataset to tune the algorithms (i.e.

select the best hyperparameters). To test the adaptability of training to unseen situations (unknown attacks), the training data for supervised learning were selected from *IntegrityMin* and *IntegrityMax* attacks on process variables (XMEAS). We generated attack samples, runs containing integrity attacks (IntegrityMin or InetegrityMax) with an attack duration ranging from 20 minutes to 3 hours. We generated 2 IntegrityMin attacks and 2 IntegrityMax attacks on each of 16 sensors that could be attacked, at random hours, to generate the training set. Four attacks brought the plant down, and therefore they were removed from the training set. In total, these 60 samples (simulation runs) containing attacks were used in training. To validate the model, another 20 runs containing attacks were used. Some initial experiments were carried out using both Decision Tree and Random Forest, and the results showed false positive rates were high. To ameliorate this problem, 12 sets of normal plant runs were added to the training data.

The test dataset that was used to measure the performance of the supervised learning methods was the same for both supervised learning and unsupervised learning. As illustrated in Table 7.2, the dataset contains 224 attack cases, comprised of integrity, DoS and replay attacks not seen during training.

7.2.2 Dataset for Unsupervised Learning

The dataset used for training the unsupervised learning methods contains 30 normal runs of the plant; another 10 runs were used to validate the model during learning. Initial experiments on One-Class SVM showed that small training datasets produced a high number of false positives, and that adding more data had no significant influence on the accuracy of results, but did increase the training time, which was already long.

7.2.3 Preprocessing Data

Standard machine learning algorithms assume that samples are independently and identically (iid) drawn from some joint distribution; however, this is often not the case with time series data like that from the TE model. The data contains significant sequential correlation and, as a result, it needs to be converted into individual

feature vectors before applying machine learning methods. There are a number of methods that have been proposed for this task, including sliding window, recurrent sliding windows, Markov models and transformer networks [306]. Based on the data, the most flexible method was to use a sliding window as it is agnostic to the learning method, and this is the method most commonly used in past studies [141]. A sliding window is created by moving a fixed size window w across the training data: $(k_t, ...k_{t+w-1})$ where $k_t = (u_{t,1}, u_{t,2}, u_{t,i}, y_{t,1}, y_{t,2}, ..y_{t,j})^T$ and where the *us* and *ys* are variable measurements and manipulated variables, and i and j are the dimensions of the variables. The classifier operates on these resulting feature vectors. Given the size of the dataset and the limited computation budget and storage, the studied size of w was in the range of [1,10].

For SVM methods and neural networks, all training, validation and testing data is normalised. All features were scaled using the *min-max* scaling method such that the maximum value is 1 and the minimum value is 0 based on the training data, using the following equation:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{7.3}$$

Classifiers using decision trees require no such normalisation.

7.3 Evaluation Metrics

The performance of the detection methods was measured using the standard metrics of *accuracy*, *precision*, *recall*, *F1 score*, and *false positive rate*, defined in Section 3.1.1. Using these metrics, the performance of test dataset was measured based on the pre-attack data and the duration of the attack. The length of the pre-attack interval is defined as the length of the attack. So, if a DoS attack is started at hour 8 for a duration of two hours, these measurements are calculated using the data from hour 6 to hour 10. The signal produced from hour 6 to 8 is labelled as not-attack, and the signal labelled from 8-10 as attack. The duration after-attack is ignored to avoid any contamination. However, to cater for detection after the end of an attack, another metric is required, which we will explain in the following section.

7.3.1 Declaring Attack

Declaring that an attack is taking place at present requires the detection to be robust to false positives, which are caused by noise in the normal operation of the plant. To cater for this, we use a sliding window to declare that an attack is present only if the number of anomalous data points in a window exceeds a threshold. The length of the window will be selected based on the false positive rates determined in normal plant operation.

Detection latency should be a key metric of anomaly detection systems; however, it is often excluded. As a result, we introduce another metric to identify when a detection takes place: during the attack or after it. The after-attack period is defined to be of a period of length equal to the attack duration; so, for a 30 minute attack, the after-attack period is the 30 minutes after the attack ended. Based on this, the attack detection rate during and post attack is calculated:

$$\text{Attack Detection Accuracy} = \frac{\text{detected attacks}}{\text{total number of attacks}} \tag{7.4}$$

7.4 Hyperparameter Optimisation

7.4.1 Hyperparameter Tuning for Supervised Learning

All learning algorithms require some tuning to determine the values of the parameters needed to optimise the performance of the learning. These parameters, which are set before the learning process begins, are called *hyperparameters*. Hyperparameter tuning is critical in machine learning as different parameters may result in models with significantly different performance. Grid search was used to determine the hyperparameters. For Decision Tree (CART) learning, the most important parameter was the maximum depth of the tree. For assembly methods using CART (Random Forest, Adaboost and Bagging) the critical parameter is the number of estimators used, which defines the number of trees in the forest or the estimators. We did not have the resources to carry out a grid search for SVM, because it took just over 48 hours to train a single model. As a result, the default parameters were used for the SVM classifier. Table 7.3 reports the parameters used for the supervised

Algorithm	Parameters
Decision Tree	Max Tree Depth = 50
Random Forest	Max Tree Depth = 100, Number of Estimators= 25
Bagging	Max Tree Depth = 100, Number of Estimators= 100
AdaBoost	Max Tree Depth = 100, Number of Estimators= 50
SVM	Kernel: RBF, C and γ=(default=auto)
One-Class SVM	Kernel = RBF, $\upsilon = 0.00346$ and $\gamma = 0.018$

Table 7.3: Parameters for learning algorithms

learning techniques.

During hyperparameter tuning, we also investigated the effect of using the sliding window to generate the feature vectors, as discussed in Section 7.2.3. We decided to operate on a window of size 1, meaning the learning algorithm operates on each time step separately. Given the dimensionality of our data (50) the sliding window rearranges consecutive time points into a single vector of size $50w$ using the entire training data. This increases the size of the data further, making it unsuitable for use with resource-hungry SVM algorithms. Increasing the window size had only a small impact on the performance our algorithms, and we did not want to slow the performance of our evolutionary algorithms further when later evolving a large population of attacks against the IDS. However, we created the effect of the window size on algorithms using decision trees, and studied a smaller version of our dataset to study the effect on SVM methods. We report these results for future reference.

Table 7.4 shows the performance of the Decision Tree and Random Forest methods. The whole training data was used to train the model. The results obtained are for the validation dataset using windows of size 1-10.

The effect of the window size on the supervised SVM was investigated using subsets of the training and validation data. Table 7.5 shows that the SVM is able to capture the correlation between the near neighbourhood datapoints using smaller windows and this improves the performance of the classifier. The performance started to decline after $window > 30$, and thus we stopped.

Decision Tree	Window Size	Accuracy	Precision	Recall	F1 score
	1	96.37	94.86	81.44	87.64
	3	96.35	94.94	81.29	87.59
	5	96.36	94.24	82.29	87.71
	10	96.37	94.09	82.25	**87.77**
Random Forest	1	96.78	95.57	83.5	89.13
	3	96.84	95.58	83.93	89.23
	5	96.80	95.59	83.66	89.37
	10	96.85	95.58	83.99	**89.41**

Table 7.4: Effects of sliding window size on the performance of the Tree Classifiers (%)

Window Size	Accuracy	Precision	Recall	F1 score
1	96.23	96.64	76.91	85.65
3	96.46	97.11	78.13	86.59
5	96.54	97.15	78.65	86.92
7	96.59	97.31	78.90	87.14
10	96.6	97.41	78.91	87.19
15	96.61	97.51	78.93	87.24
20	96.62	97.56	78.93	**87.26**
30	96.59	97.45	78.86	87.17

Table 7.5: Effects of sliding window size on the performance of the SVM (%)

7.4.2 Hyperparameter Tuning for One-Class SVM

The effects of the sliding window size on the unsupervised, One-class SVM was also examined using a subset of the training data. However, since One-Class SVMs are more sensitive to choice of parameters, this time we did investigate using a grid search to optimise the parameters. One-Class SVM has two parameters that require tuning, υ and γ. υ is the upper bound on the fraction of outliers and a lower bound of the fraction of support vectors [92]. The value of υ is tuned according to the expected outliers in the data. The second parameter γ is the kernel coefficient. The kernel used is the radial basis function (rbf). To investigate the effects of these parameters, a simple grid search was carried out. This involved exhaustively generating solution instances from a grid of parameter values. The F1 scores for the combinations of parameters are shown in Table 7.6. According to these results, the best performing parameters are around $window = 3$, $\upsilon = 0.001$ and $\gamma = 0.01$.

As stated above, given that the sliding window had no significant effect on

window=1				
$\upsilon \setminus \gamma$	10^{-4}	10^{-3}	10^{-2}	10^{-1}
10^{-4}	71.69	73.30	78.36	39.32
10^{-3}	75.48	76.18	79.37	48.39
10^{-2}	79.64	79.90	**81.35**	48.41
10^{-1}	70.50	70.54	70.75	48.42
0.2	60.69	60.70	60.71	47.94
0.5	42.78	42.77	42.74	40.53
window=3				
$\upsilon \setminus \gamma$	10^{-4}	10^{-3}	10^{-2}	10^{-1}
10^{-4}	72.77	77.13	83.35	29.48
10^{-3}	76.62	78.99	**83.35**	29.48
10^{-2}	80.58	81.36	82.81	29.48
10^{-1}	71.01	71.13	71.02	29.48
0.2	60.95	60.96	60.47	29.48
0.5	42.70	42.69	42.62	29.48
window=5				
$\upsilon \setminus \gamma$	10^{-4}	10^{-3}	10^{-2}	10^{-1}
10^{-4}	73.29	79.68	77.64	29.48
10^{-3}	77.07	80.59	77.66	29.49
10^{-2}	80.86	**81.94**	77.66	29.49
10^{-1}	71.06	71.15	69.26	29.49
0.2	61.07	61.09	59.68	29.49
0.5	42.72	42.68	42.43	29.49

Table 7.6: F1 scores for the One-Class SVM Parameters (%)

outcomes for the decision tree classifiers, and SVMs are computationally expensive for large datasets, a window size of 1 was used to ensure that the training data used for all for the experiments were consistent. As we explain later, having the same window size also helped us to compare the false positive rates for all the classifiers as accurately as possible.

Grid search was used to find the optimal parameters for One-Class SVM on the training data. The best parameters found were $\upsilon = 0.00346$ and $\gamma = 0.018$, and thus these were used to train our One-Class SVM.

7.4.3 Hyperparameter Tuning for Deep Neural Network

Training deep neural networks can be a complex task due to the number of hyper-parameters to optimise. These parameters include: deciding on the architecture and the number of layers to use; deciding on the number of neurons at each layer and

Algorithm	Network Architecture	Optimizer	Dropout	Batchsize
Autoencoder	1 hidden dense layer x 40 units	Adam	-	1024
LSTM	2 layers x 64 units	Adam	0.1	1024
GRU	2 layers x 64 units	Adam	0.1	1024

Table 7.7: Architecture for Deep Neural Networks

activation functions (e.g. softmax, relu, tanh, sigmoid, exponential, linear); and selecting an optimiser. Each optimiser has its own internal parameters to tune such as learning rates and decay functions, all of which may themselves need to be optimised to help the learning. We approached the problem using the general advice to start with just one or two hidden layers and gradually building from there using manual tuning.

The parameters used for our Deep Neural Networks are given in Table 7.7; any parameters not shown in this table use the default parameters obtained from the Keras [281] library we used to implement these networks. In the following section, we explain how these parameters were selected.

7.4.3.1 Training the Autoencoder

To remind the reader, as shown in Figure 7.3, the input values are introduced into the network using the input layer. The encoder part of the function maps the input data to a latent representation, and a decoder function attempts to transform the latent representation back to the initial input at the output layer. By limiting the number of hidden units, one is able to create a bottleneck and force the network to learn interesting structure about the data. So, by forcing the network to learn a compressed representation of the data, it learns the most important features. The parameters for the autoencoder are designed to reduce the reconstruction error. As we explained in Chapter 4, once the network is trained on normal data, it can be used to detect anomalous behaviour by using a threshold based on the reconstruction error. Since the training data does not include attack samples, the reconstruction error will be high for these cases. Thus, using a threshold based on the reconstruction error, one can try to predict whether new data is normal or abnormal (under attack).

50 units were used for the input and output layers (equal to the dimension-

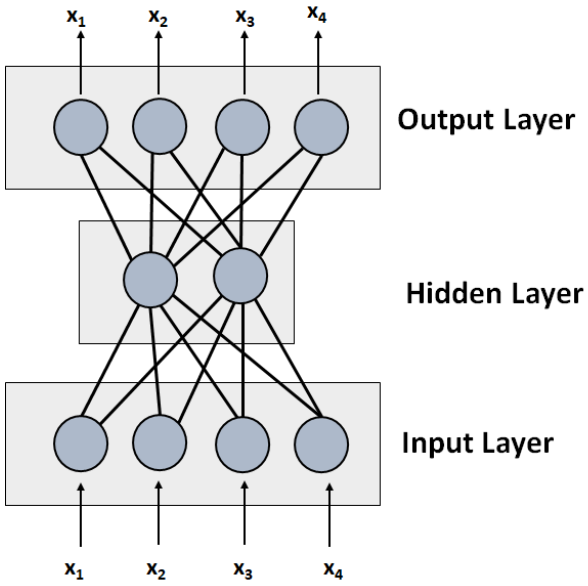


Figure 7.3: Structure of an autoencoder

ality of our data). The number of units used for the hidden layer was optimised manually. Table 7.8 shows the performance of the network using various number of hidden units tested on the test data containing the attacks. The best performance was achieved using a network with a hidden dense layer of 40 units. The network was trained using the Adam optimiser with its default parameters, and a batch size of 1024 with early stopping. Adding more hidden layers failed to improve the performance of the network but, given that the hyperparameters were empirically decided, it is possible that this is not the optimal network.

No. of Hidden Units	Accuracy	Precision	Recall	F1-score
20	94.89	91.10	65.10	71.90
36	94.40	91.11	64.2	71.25
38	94.51	90.90	65.85	72.20
40	95.07	90.34	71.01	**75.86**
41	94.93	89.86	70.59	75.12
42	94.55	90.06	66.73	72.50
48	94.01	92.07	59.84	67.45

Table 7.8: Tuning number of hidden units in the autoencoder

To predict anomalous behaviour, the reconstruction error is calculated using the mean square error (MSE), using the actual values and the output (predicted)

values (n=50 dimensions) for each time point:

$$MSE_t = \frac{1}{n}\sum_{t=1}^{n}(actual_t - predicted_t)^2 \tag{7.5}$$

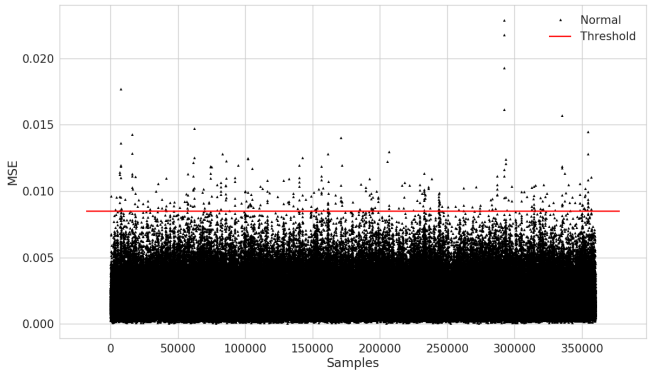


Figure 7.4: MSE error on validation dataset (under normal operating conditions)

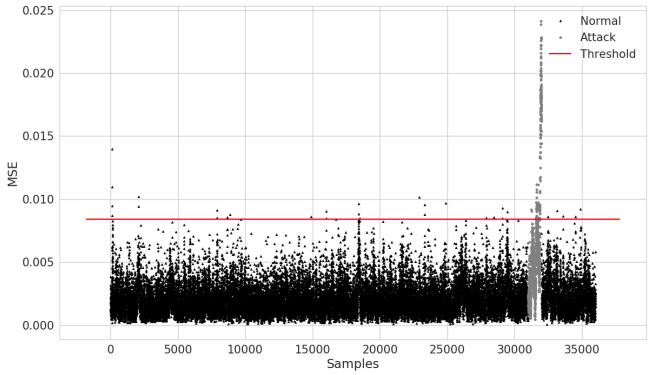


Figure 7.5: MSE values for a replay attack started at Hour 62 (around 31000 data samples)

If the error calculated at the time point t is larger than the defined anomaly threshold, then it is classified as an attack. The anomaly detection threshold value is calculated as the 99.90% confidence interval of the MSE error distributions on the validation dataset that we used to tune our learning. This threshold was then used to evaluate the unseen attacks in the test dataset. Figure 7.6 shows the MSE values for the validation dataset, and the red horizontal line shows the defined threshold, which is 0.008478. Points above the line are classified as attack, and anything below the

line is considered not-attack. Figure 7.5 shows the MSE values for the plant being attacked around hour 62.

7.4.3.2 Training Recurrent Neural Networks

We adapted the LSTM architecture proposed in [123] to implement the architecture for the recurrent neural networks: GRU and LSTM. As indicated in Table 7.7, these network used two consecutive hidden layers with 64 units. GRU networks are simpler than LSTM and they tend to train faster. For both hidden layers, *tanh* activation functions were used with a dropout rate of 0.1 to prevent overfitting. The output layer is a dense layer using the *linear* activation function. The number of neurons in this output layer is equal to the number of dimensions in the data, 50, with one unit for each feature predicted. The *lookback* and *lookahed* parameters are the number of past steps used by the RNN to predict the number of steps in future. The optimal value for these windows depends on the data and the correlations within the time series. However, it is usually the case that the larger the forecast window, the harder the problem. We made the problem simpler, but slower, by predicting one single time step at a time.

Window Size	Accuracy	Precision	Recall	F1 score
3	80.90	83.47	51.17	58.13
5	81.01	83.34	51.46	58.34
10	81.1	83.2	51.80	58.70
20	80.58	83.62	51.58	58.41
50	80.71	83.68	51.90	58.72
70	80.64	84.16	52.25	59.06
100	80.55	84.38	52.61	59.52
120	81.00	85.11	54.65	**61.22**
150	78.65	85.68	46.86	55.05

Table 7.9: Effects of look-back window size on the performance of the GRU (%)

Table 7.9 and Table 7.10 show the performance of the various sized *lookback* windows; this shows that the best look-back size is 120. Our experiments showed that increasing the look-back window size improves the prediction performance up to a limit, and window sizes over 120 produced no significant improvement on the performance. Both RNNs were trained using the *Adam* optimiser with its default

Window Size	Accuracy	Precision	Recall	F1 score
3	80.89	83.53	51.17	58.18
5	80.58	83.54	51.86	58.79
10	80.65	83.58	50.21	57.56
20	80.82	83.24	51.49	58.54
50	80.56	83.31	51.03	58.10
70	80.38	83.42	50.98	58.03
100	80.26	83.75	51.67	58.56
120	80.51	84.18	52.40	**59.23**
150	79.51	83.85	50.67	57.66

Table 7.10: Effects of look-back window size on the performance of the LSTM (%)

parameters, and a batch size of 1024 with early stopping. Figure 7.6 shows the model loss for the LSTM and GRU during training. The LSTM model performed slightly better than GRU, achieving a slightly lower model loss.

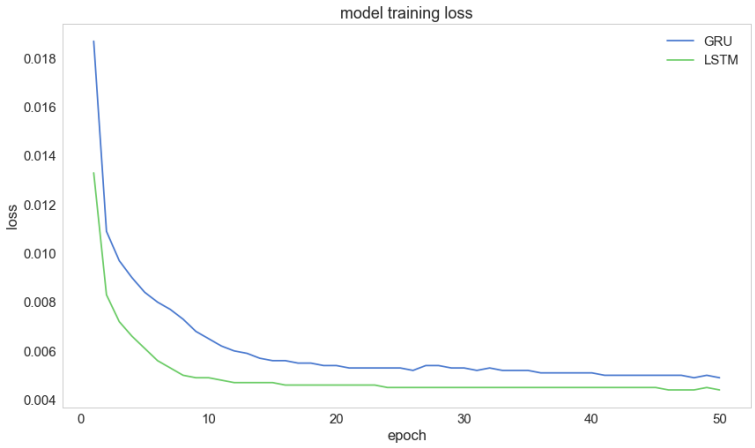


Figure 7.6: Comparison of LSTM and GRU model loss

7.5 Results and Analysis

In the following sections, we report the performance of our detection methods.

7.5.1 Supervised Learning

Table 7.11 shows the performance metrics for classifiers using supervised learning for all three types of attacks, and also the overall statistics.

	Decision Tree	Random Forest	AdaBoost	Bagging	SVM
All					
Precision	82.61	87.91	77.75	82.91	81.81
Recall	54.89	59.08	35.65	56.55	51.37
F1 score	60.97	**65.45**	42.46	62.42	58.63
Integrity					
Precision	95.09	96.10	98.04	95.78	99.07
Recall	83.13	86.87	62.87	84.79	76.17
F1 score	87.74	90.54	72.40	88.97	84.40
DoS					
Precision	84.36	88.72	73.21	85.06	82.14
Recall	48.56	51.95	26.64	50.68	45.03
F1 score	56.63	60.60	33.29	58.42	52.75
Replay					
Precision	66.43	77.54	56.83	65.97	60.96
Recall	25.75	31.08	9.68	27.08	26.35
F1 score	32.15	39.00	13.37	33.59	32.11
FPR	0.19	0.15	0.016	0.14	**0.0067**

Table 7.11: Detection performance comparison for supervised learning (%)

All classifiers achieved their best result for the integrity attacks, and failed to show the same detection ability on previously unseen attacks. The results show that using using multiple decision trees (predictors) is better than using a single decision tree, as the performance of the Bagging and Random Forest are better than single Decision Tree. As explained in Chapter 4, Random Forest is an ensemble of decision trees formed using the bagging method, and improves on the Decision Tree (CART algorithm) method by growing more diverse trees. Decision Tree construction considers all features when splitting a node whereas Random Forest searches for the best feature from a subset of features selected at random. Boosting (AdaBoost) had the worst F1 score of all the decision trees, Bagging and Random Forest. There has been a number of studies using fault detection and diagnosis on the

TE model using Boosting with decision trees, and the logical next step was to investigate whether there were any benefits in using Boosting CART. This proves not to be effective. Overall, Random Forest showed the best F1 performance; however, it comes at a cost, with higher false positive rates than SVM and Boosting.

The SVM achieved the best false positive rates, at a rate of 0.0067% but it also had a lower detection performance, and a wider gap between precision and recall, as reflected in F1 score. Moreover, SVM is not suitable for large data. Tree-based classifiers deal better with large datasets, and they are faster to train and test than an SVM. The training and validation of the Random Forest took just under an hour, whereas SVM took over 48 hours. Furthermore, Random Forest can be trained in parallel to improve the training performance even further. This is particularly important when searching the space for the optimal hyperparameters. Considering the speed, accuracy and complexity, the best supervised learning method for this particular problem is Random Forest.

7.5.2 Unsupervised Learning

Table 7.12 presents a comparison of the four unsupervised learning techniques evaluated.

All unsupervised techniques performed better at detecting the replay attacks than supervised techniques; however, their performance on integrity attacks was worse than supervised methods. One-Class SVM and LSTM did worse on DoS attacks compared to Random Forest. Overall Autoencoder had the best performance with a slightly better precision and recall than One-Class SVM. Autoencoder was better than One-Class SVM at detecting DoS and replay attacks but peformed worse on the integrity attack. The overall F1 score for these methods was better than supervised learning methods because they were better at detecting DoS and replay attacks. On average, LSTM and GRU performed worse than One-Class SVM and Autoencoder. However, for DoS attacks, their performance was better than One-Class SVM, indicating that no single algorithm might give the optimal performance for all possible types of attack. The GRU network's performance was better than LSTM but it came at a slightly higher cost of false positives.

Overall:	One-Class SVM	Autoencoder	LSTM	GRU
Overall **All**				
Precision	89.30	89.34	84.18	85.11
Recall	63.40	63.83	52.40	54.65
F1score	68.69	**69.95**	59.23	61.22
Integrity				
Precision	92.19	91.84	85.43	86.08
Recall	83.18	75.91	66.06	69.55
F1score	**86.62**	81.36	72.27	75.74
DoS				
Precision	87.94	92.65	87.09	89.08
Recall	50.20	66.05	53.46	53.98
F1score	57.01	**72.89**	60.54	61.14
Replay				
Precision	84.76	83.94	80.56	81.06
Recall	41.42	47.78	36.76	37.33
F1score	47.61	**54.18**	42.68	43.93
FPR	0.44	**0.10**	0.90	0.99

Table 7.12: Detection performance comparison for unsupervised learning (%)

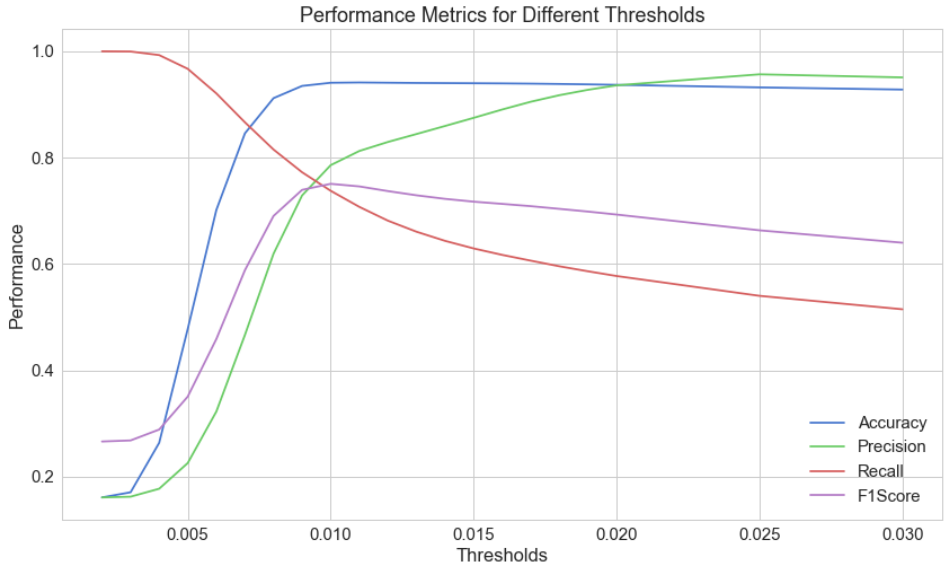


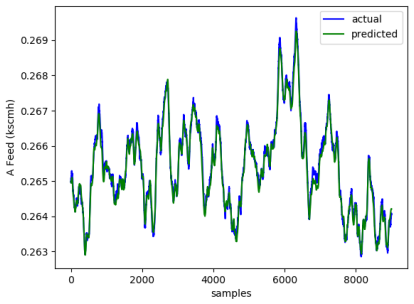
Figure 7.7: Performance of LSTM for different MSE thresholds

Figure 7.7 shows the performance of the LSTM network using various MSE thresholds on the validation dataset. The threshold can be tuned further to increase the F1 score and get the right balance between recall and precision. The lower

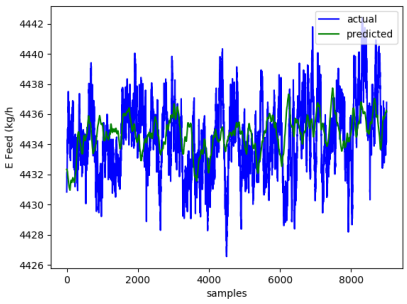
threshold used for this case was 0.01387 (99.90 % percentile of the error distribution). However, the performance measures showed that by using a lower threshold, say 0.010, it is possible to achieve a higher F1 score of 75.07% with a more balanced precision rate of 78.58% and a recall rate of 72.76%. Handling too many false positives can be very expensive, and thus organisations will need to tune this parameter based on their security policy. A more in-depth comparison of false positive rates is given in the following section, which looks at the ability of the detectors to tell whether an attack is taking place.

7.5.2.1 Accuracy of RNN Predictions

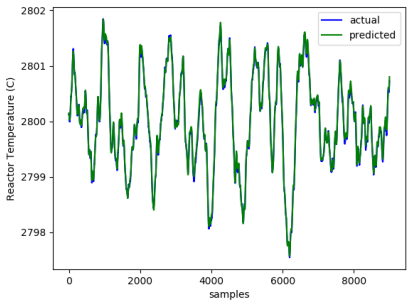
LSTM and GRU show that not all features in the timeseries give the same prediction accuracy: some features are easier to forecast than others. Figure 7.8 shows four of the features, A Feed, E Feed, Reactor Pressure and Separator Underflow.



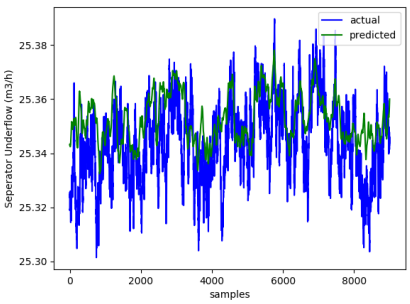
(a) A Feed



(b) E Feed



(c) Reactor Temperature



(d) Separator Underflow

Figure 7.8: Example of a LSTM prediction

Predicting the E Feed or Separator Underflow is harder than predicting the values for Reactor Temperature and A Feed. The MSE used to identify what is

normal behaviour uses a combined score for all features. This might not be the best option for multivariate non-linear time series data with noisy forecasts. As a result, future work should examine this and investigate a better technique for calculating the errors between the predictions and actual data. To achieve better MSE scores, we suggest two options. The simpler option is to eliminate the features that are difficult to predict, and use a subset of the features. The second option is to give weights to features when calculating the MSE values so that features that are easier to predict are have higher weights than those that are hard to predict.

7.5.3 Dealing with False Positive Rates and Detecting Attacks

All classifiers have some noise in the detection that manifests as false positives. In a real physical system, there will probably be even more noise than that seen in these simulations due to behaviour of the physical components of the systems (e.g. actuator and sensors degrading over time, components of the plant wearing, or other kinds of natural noise in the environment). Detection should be robust against natural noise, and distinguish between the normal plant disturbances and attack conditions.

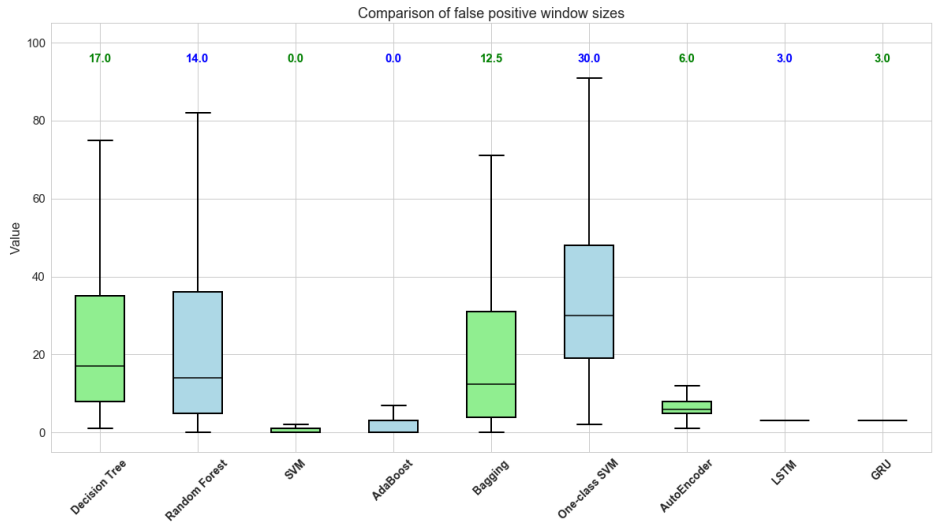


Figure 7.9: Percentage of false positives for detection models in a window size of 100 (under normal operating conditions)

To achieve this, a sliding window of size 100 was used to study the percentage

of false positives under normal operating conditions. We run the plant 1000 times under normal conditions, without any attacks, using a different random seed for each replicate to ensure that randomness was achieved. Then, using each detection method, we investigated the highest false positive percentage seen for each run. The results show an interesting pattern, illustrated in Figure 7.9. The false positive rate for LSTM was 0.90%; this means that when the plant is running under normal condition and produces 36000 datapoints, about 324 (0.009 x 36000) of those points are false positives. As the boxplots in Figure 7.9 show the range is not large. This is further investigated in Figure 7.10, these are not long sequences of consecutive false positives. Smoothing these errors using an exponentially weighted moving average (EWMA), as shown in Figure 7.10, could improve the false positive, but this means a slight delay in anomaly detection.

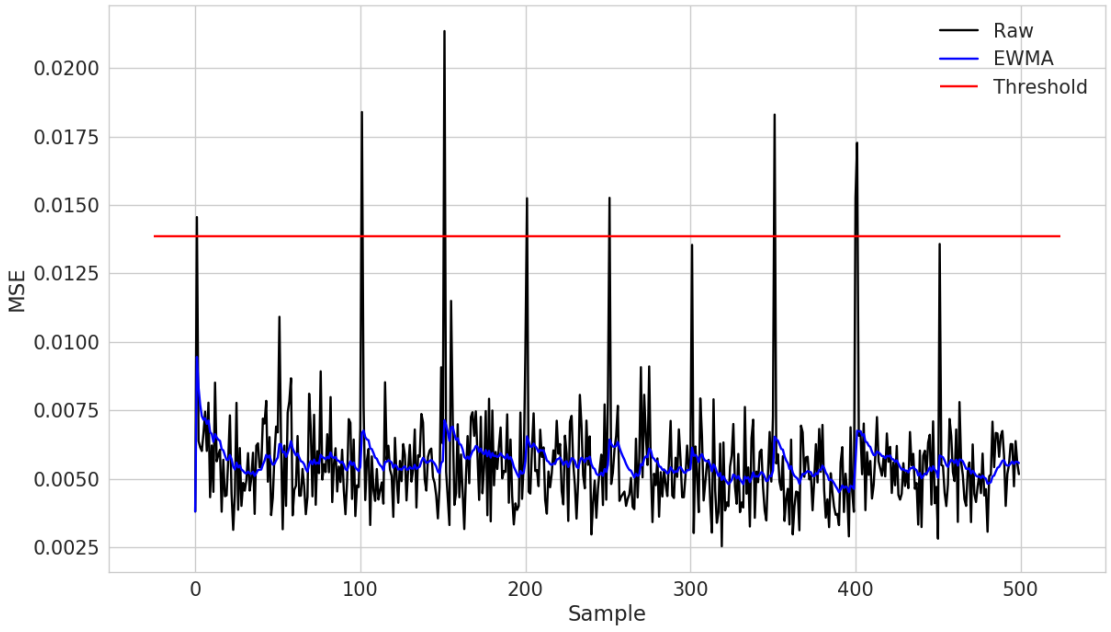


Figure 7.10: EWMA smoothing to reduce false positives

The proportion of false positives for the Decision Tree was 0.19%, about 68 (0.0019 x 36000) points per simulation. However, the boxplot in Figure 7.9 shows the range for the percentage of false positive probabilities is wider. This is also the case for Random Forest, Bagging and One-Class SVM, but RNNs, SVM, Autoen-

coder and AdaBoost have a smaller range. This is an advantage because a smaller sliding window can be used to detect the presence of attacks. The data sampling rate is 0.12 minutes per point, so a window of size 100 is equal to 12 minutes. The thresholds shown in Table 7.13 were designed to determine the presence of the attack. For example, SVM only declares an attack if 50% of the data points in the sliding window is classified as attack, giving a minimum detection time of 6 minutes.

Algorithm	Threshold (%)
Decision Tree	100
Random Forest	100
Bagging	100
AdaBoost	50
SVM	50
One Class SVM	100
Autoencoder	35
LSTM	6
GRU	20

Table 7.13: Attack detection thresholds

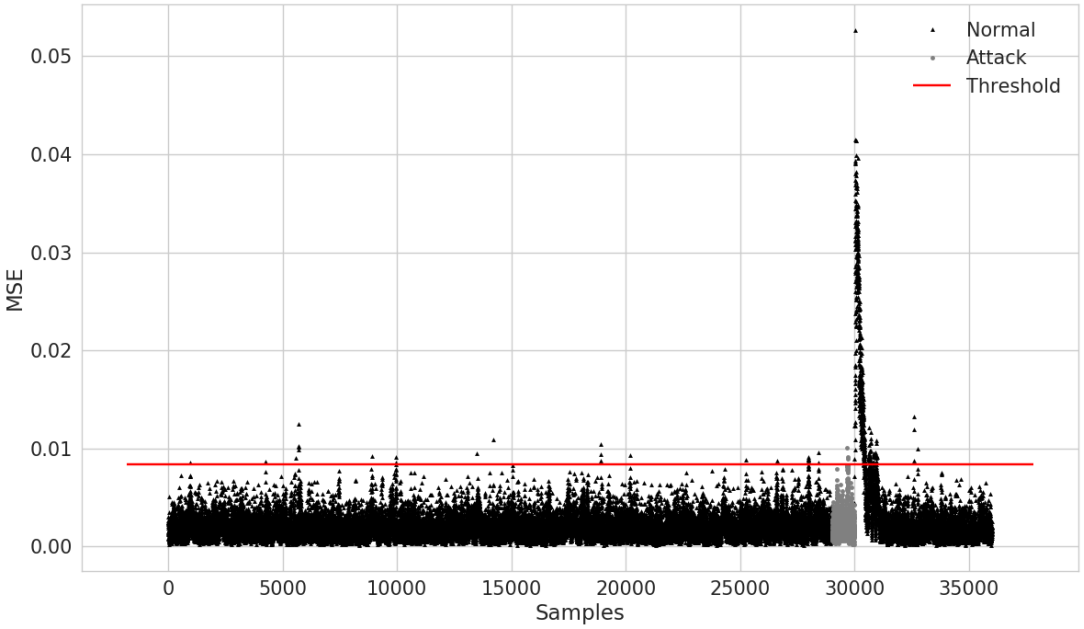


Figure 7.11: Example of an attack detected after it has stopped

As we discussed earlier, in carrying out a variety of attacks, we observed that the anomalous behaviour is almost never equal to the attack interval, and it is possible that some attacks will be detected after the attack interval if the anomalous behaviour persists. Figure 7.11 shows one of these attacks, a replay attack detected by the autoencoder after the attack interval. Therefore, we decided to use a window equal to the attack interval after the attack to investigate the attacks detected.

Using the attack detection thresholds in Table 7.13 and the number of attacks detected during the attack and post attack periods, we calculated the attack detection rates. Table 7.14 and 7.15 shows the attack detection probabilities for supervised learning and unsupervised learning. For supervised learning, SVM detected all the integrity attacks when the attacks were taking place. The Decision Tree, Random Forest and Bagging methods achieved the highest DoS detection rate, 75%; this score was achieved using the detection interval after the attacks stopped. Random Forest and Bagging did better than the other classifiers on detecting the replay attacks but, nevertheless, they were not as good as the unsupervised techniques.

Detector	Decision Tree		Random Forest		AdaBoost		Bagging		SVM	
	During	After	During	After	During	After	During	After	During	After
Integrity	97.80	98.91	94.57	98.91	88.04	93.48	95.65	98.91	**100.00**	100.00
DoS	66.07	**75.00**	62.05	**75.00**	44.64	51.79	67.86	**75.00**	71.43	71.43
Replay	33.77	53.25	40.26	**55.84**	19.48	35.06	40.26	**55.84**	36.25	53.25

Table 7.14: Attack detection probabilities for supervised learning (%)

Detector	One-Class SVM		Autoencoder		LSTM		GRU	
	During	After	During	After	During	After	During	After
Integrity	96.74	96.74	95.65	95.65	**97.83**	97.83	95.65	95.65
DoS	78.57	80.36	87.50	**90.74**	80.36	83.93	78.57	80.36
Replay	53.27	58.44	62.34	**67.53**	63.63	66.23	61.04	63.64

Table 7.15: Attack detection probabilities for unsupervised learning (%)

7.6 Summary

In this chapter, we presented an extensive investigation of some of the classic machine learning techniques and some new deep learning techniques to answer Re-

search Question 3 (How should one design a novel intrusion detection system to detect attacks?). Somewhat surprisingly, unsupervised techniques appear to be more promising than supervised techniques for detecting attacks. The supervised learning algorithms performed better on the trained integrity attacks but failed to show similar performance on the DoS and replay attacks for which they were not trained.

Unsupervised learning techniques are the subject of increasing research interest, but the question of how to apply these techniques to ICS environment is an area that requires further research. The techniques evaluated in this chapter show promise for further study: they were able to raise more alarms, especially for DoS and replay attacks. Using a single hidden layer, the autoencoder did better than the One-Class SVM and RNNs. On average, RNNs did worse than One-Class SVM but their performance on DoS was better. There were no methods that scored highest on all three types of attacks; consequently, a single detection method may not be suitable for detecting all types of attacks. Autoencoders are often used with LSTM to detect anomalous behaviour in time series data, making use of the correlation in the sequences of the data. There is much more work to be done in this area, and future research will look into this. Aside from a presence indication, proactive detection and mitigation of anomalous behaviour in ICS requires two further steps:

- Diagnosis: Identify the cause of the anomaly
- Recovery: Take the necessary measures to return the system to normal operating conditions.

Future research will investigate how to achieve these steps using supervised and unsupervised learning techniques. Although fault diagnosis is not feasible with classic unsupervised learning techniques such as One-class SVM, neural networks might offer some guidance by utilising the MSE errors. For example, identifying the features that contribute the greatest amount to the MSE could potentially help with the diagnosis.

The existing performance metrics that we used to measure the performance of the detection are insufficient on their own for ICS as they do not cater for detection

latency. The community needs to develop generally accepted performance metrics that encompass this dimension.

In the next chapter, we evolve attacks against some of the detection models we implemented in this chapter using the evolutionary multiobjective optimisation approach, to further investigate their performance.

Chapter 8

Evolving Attacks Against the Intrusion Detection System

In this chapter, we investigate Research Question 4: Can one evolve new attacks against the Intrusion Detection System? We adapt the approach developed in Chapter 6, using evolutionary multiobjective optimisation and set it against the intrusion detection mechanisms we developed in Chapter 7. Our objective is to generate attacks that evade detection while causing some damage.

We first explain how we formulated the research problem, define the objectives, and explain the algorithmic approach. Finally, we report the results and our analysis.

8.1 Evading Detection using Evolutionary Multiobjective Optimisation Approach

In this section, we explain the objectives of our evolutionary multiobjective optimisation, and outline the steps involved in generating attacks that evade detection. For this we considered three common objectives that both the adversary and defence need to consider. These are *maximise damage*; *minimise detection probability*; and *minimise effort* required to carry out the attacks:

$$F(x) = [f1(max), f2(minimise), f3(minimise)] \tag{8.1}$$

where $f1$ is to *maximise damage* in terms of increasing the operating cost of

the plant, $f2$ is to *minimise detection probability*, and $f3$ is to *minimise the effort* required to carry out the attack:

$f1$ = Difference between the cost of the plant under attack and the operating cost of the plant under normal operation (the damage caused by an attack).
$f2$ = Detection probability for the attack, denoted as a percentage.
$f3$ = Total number of sensors and actuators attacked (the effort required to launch an attack).

We generated attacks using two objectives (f1, f2) and three objectives (f1, f2, f3), to understand whether minimisation of effort had a significant effect on damage and detection.

The performance of anomaly-based detection for both supervised and unsupervised learning showed that the integrity attacks carried out on both sensors and actuators were detected with high accuracy; however, the performance of DoS and replay attacks failed to achieve similar detection probabilities. For this reason, we decided to use only DoS and replay attacks as a basis for the evolutionary algorithms.

Intrusion detection was developed using supervised classifiers, AdaBoost, Decision Tree and Random Forest, and an unsupervised classifier, One-Class SVM. AdaBoost has previously been used in Network Intrusion Detection, e.g. by Hu et al. [307], but represents an approach that is currently not at the leading edge of machine learning. Decision Tree, Random Forest and One-Class SVM are models that are more frequently used, but these are selected both because they represent a situation in which a system's defences has become somewhat outdated, and they do not require special computational resources.

8.1.1 Evolutionary Multiobjective Optimisation Algorithm

Algorithm 7 illustrates the steps involved in generating attacks against the detection models. This is same as the previous Algorithm 6 that was developed in Section 6.2.1 except that this time user can specify the selection strategy. The flow chart

shown in Figure 8.1 shows the structural components of the evolutionary multiobjective optimisation for generating attacks against the detection. Table 8.1 shows the evolutionary operators and the parameters used to evolve attacks against detection. In the following section, we discuss the details of the approach.

Parameters	Value
Chromosome size	25
Types of allele for each chromosome	140
Description of alleles	Do not Attack, DoS, Replay with various starting time
Number of generations	500-1000
Parent population size (μ)	200
Offspring populations size (λ)	300
Crossover probability	0.85
Mutation probability	0.1
Probability of mutating a gene in a chromosome	0.05
Crossover operator	Two-point crossover
Mutation operator	Uniform mutation
Selection operator	NSGA-II, SPEA2
Selection strategy	$(\mu,\lambda),(\mu+\lambda)$
Objectives	Maximise f1, Minimise f2, Minimise f3 f1=damage caused, f2=detection probability f3=attack effort
Parallel evaluations	50
Computation time	$>$70 hrs

Table 8.1: Evolutionary operators and parameters for generating attacks against detection using EMO

Chromosome Encoding. As before, individuals are represented as chromosomes encoded as a list of 25 integers (genes). Position of the gene (locus) denoting a sensor or an actuator. To make the problem computationally tractable, we limited the types of genes to a pool of 140 in which half denoted DoS attacks and the remaining half denoted replay attacks. Each number type denotes the form and the start time of the attack. The duration of the attacks was kept constant for all attacks, $\leq$ 2 hours. Increasing the duration of attacks increases damage on the plant, but it also increases the risk of detection, consequently, we utilised short duration attacks. This is a combinatorial search problem of size 140^{25}.

Algorithm 7: Evolutionary multiobjective optimisation algorithm for generating attacks against detection

Input : μ = parents to select for next generation,
λ = offspring to produce at each generation,
$mutpb$ = mutation rate, $cxpb$ = crossover rate,
$ngen$ = number of generation,
$selES$ = selection strategy [(μ+λ), (μ,λ)],
$selOp$ = selection operators [$NSGAII$, $SPEA2$]

Output: Pareto optimal front

```
1  Function Main(μ, λ, mutp, cxpb):
2      ParetoFront = []
3      pop = generate initial population randomly
4      pop = evaluate(pop)
5      gen = 1
6      selectionpop = []
7      while gen ≤ ngens do
8          offspring = vary(pop,λ, cxpb, mutpb)
9          offspring = evaluate(offspring)
10         if selES is (μ+λ) then
11             selectionpop = pop + offspring
12         else if selES is (μ,λ) then
13             selectionpop = offspring
14         if selOp is NSGAII then
15             pop = selectNSGAII(selectionpop, μ)
16         else if selOp is SPEA2 then
17             pop = selectSPEA2(selectionpop, μ)
18         ParetoFront.update(pop)
19         gen = gen + 1
20     return ParetoFront
21
22 Function vary(pop, λ, cxpb, mutp):
23     offspring=[]
24     for i = 0 to lambda do
25         random = randomGenerator(0, 1)
26         if random < cxpb then
27             parent1, parent2 = randomlySelectTwoParents(pop)
28             child1, child2 = crossover(parent1, parent2)
29             offspring.add(child1)
30         else if random < cxpb+mutp then
31             child = randomlySelectParent(pop)
32             child = mutation(child)
33             offspring.add(child)
34         else
35             parent = selectParentRandomly(pop)
36             offspring.add( parent)
37     return offspring
```

Initialisation. Given the high detection probability, we decided to generate the initial parent population of chromosomes of size of size $\mu = 200$ randomly, as previously, but limited the number of attacks in each chromosome to 1-5 attacks to speed up the evolution. These chromosomes are the initial population$_0$. The fitness of these chromosomes is evaluated by converting the chromosomes into MATLAB scripts that can be executed on the TE process. A pool of MATLAB jobs was assigned to evaluate the fitness of these chromosomes. As illustrated in the flow diagram, Figure 8.1, the MATLAB jobs take the assigned individuals (attack scripts), runs each individual on the TE simulation model, and then use the output data from the TE model to call the IDS determine the detection probability for each individual. The operating cost of the plant, plant operating time, and detection probability are written to a file, from which the EMO can collect the fitness of the individuals. To remind the reader, the detection mechanism uses a window of size 100; that is, it looks at the previous 100 data points, and reports the number of attacks in this window as a percentage.

Evolutionary Loop (Iterations). Once the fitness of the chromosomes in the initial population has been established, the evolutionary loop generates the next generation population, as illustrated in Algorithm 7. First, the chromosomes in the initial population are subject to the genetic variation operators to generate the next population of offspring of size λ. This is shown as the call to the *vary* function in Algorithm 7 in which on each of the λ iterations, randomly chosen chromosomes are subject to one of the three operations: two point crossover (with a probability of *cxpb*), uniform mutation (with a probability of (*mutpb*), or reproduction. As a result, the function vary takes an population of μ and expands it to an offspring population of size λ. Next, the EMO uses the selection strategy to decide which chromosomes should be used for the selection. If the (μ,λ) strategy is selected, then the next generation of the population is produced from only from the λ offspring just generated. If the (μ+λ) strategy is selected, then the next generation of the population is produced from both the generated offspring and the parent. In the experiments carried out in Chapter 6, the (μ,λ) strategy was used to avoid premature convergence;

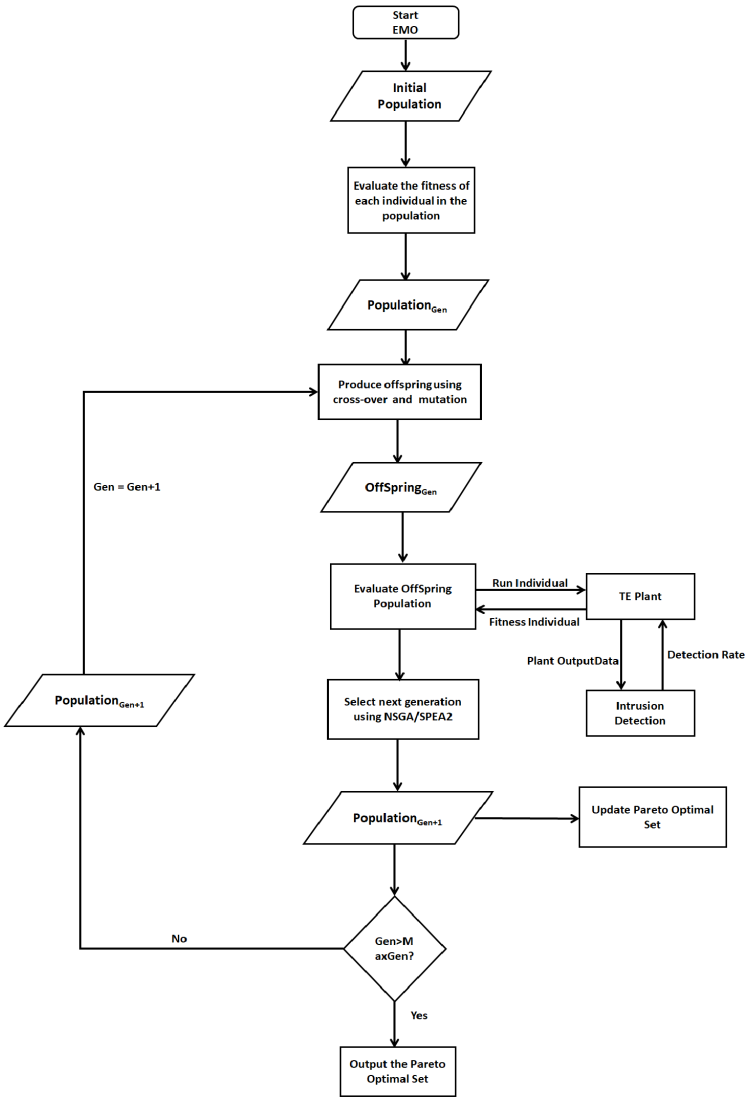


Figure 8.1: Flow diagram of the EMO-based approach designed to generate attacks against detection

here, we report and discuss results obtained for both strategies. The next population is selected using the selection methods of NSGA-II or SPEA2 and the generation counter is incremented. The Pareto front set is updated with results from the new population, and we use the elements in this set to calculate the hypervolume.

Termination Condition. The evolution process continues until either a defined number of generations has been reached, or until we exceed our maximum time allocation limit on the high performance computing platform.

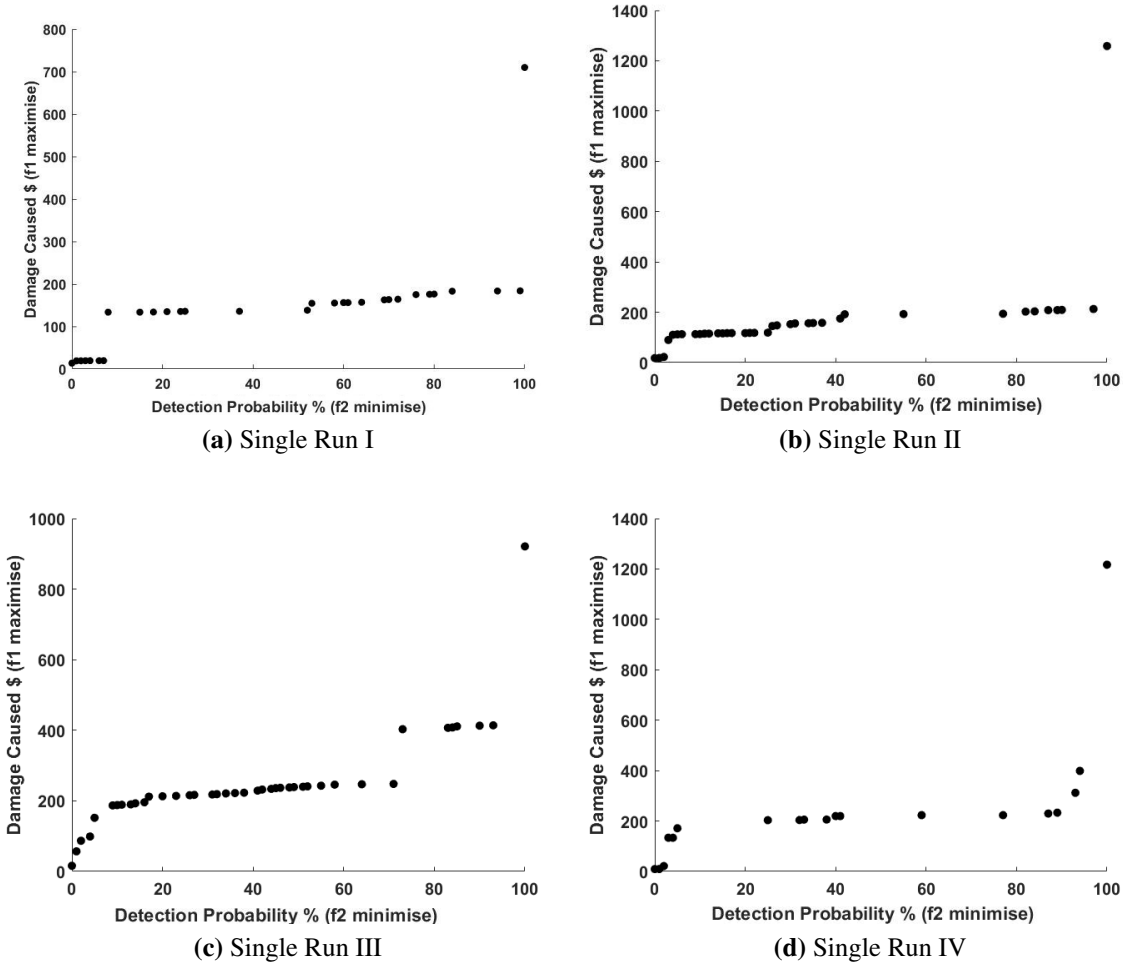
8.2 Results and Analysis

In the following section, we report the results obtained for generating attacks against the supervised (AdaBoost, Decision Tree and Random Forest classifiers) and unsupervised (One-Class SVM) intrusion detection models.

8.2.1 Generating 2-Objective Attacks against AdaBoost

Figure 8.2 presents the optimal set of individuals for the two objective optimisation ($f1$, $f2$) obtained from four different attack generation runs against the AdaBoost classifier using the genetic parameters in Table 8.1 with the (μ+λ) strategy and over 500 generations. Figure 8.2 shows that the EMOs were able to find a range of attacks for the two objectives, damage and detection probability. As compared to integrity attacks, the DoS and replay attacks have less impact on the operating cost. The duration for these attacks was 2 hours and, according to these runs, the generated attacks were able to increase the cost by, at the very best, just under \$1400. As discussed in Chapter 7, the operator of the plant will need to decide on a threshold for an alarm based on the detection probability. This is a trade-off between actual attacks and false positives that are caused by randomness in the plant. Assuming that the operator chooses a very low value for the threshold such as threshold $<= 5\%$, the damage caused is around \$150, as shown in Figure 8.2c and Figure 8.2d. The average operating cost of the plant was \$8208.

Although formulating the problem as a 2-objective optimisation problem can help to determine the range of the attacks the intrusion detection is able to detect, it does not tell us anything about the effort required to carry out the attack. This is an important parameter both for the adversary and the defence when making decisions. One of the attack strategies generated is illustrated in Table 8.2. The numbers in the brackets show the attack start time; that is, an attack denoted as $\text{XMEAS15}_{\text{REPLAY(70)}}$ means a replay attack is carried out on XMEAS 15 starting at hour 70. This attack therefore involves attacking a total of 11 sensors and actuators. It might be the case that not all of these sensors and actuators are required to achieve this impact, and it might also be possible that, even by attacking fewer sensors and actuators, one can not reduce the detection probability further. To investigate



(a) Single Run I

(b) Single Run II

(c) Single Run III

(d) Single Run IV

Figure 8.2: Pareto front of attacks generated against AdaBoost (2-Objective optimisation)

these issues further, we decided formulate the problem as a 3-objective optimisation problem.

8.2.2 Generating 3-Objective Attacks against AdaBoost

In order to determine the most effective selection strategy to use, we tested the 3-objective optimisation using both (μ+λ) and (μ,λ) strategies. Figure 8.3 shows the performance of the two selection strategies used with NSGA-II and SPEA2's selection operator. These strategies select the set of individuals that compete in the selection of offspring. Table 8.3 shows the performance metrics of the two selection strategies used with NSGA-II and SPEA2's selection operators, averaged over 2 runs, for 800 generations. Both runs ideally required more time to converge, but,

Chromosome	0, 0, 0, 0, 70, 70, 30, 70, 70, 0, 0, 0, 70, 0, 0, 35, 0, 35, 0, 15, 65, 0, 0, 0, 70
Description	$XMEAS1_{NoAttack}$,$XMEAS2_{NoAttack}$,$XMEAS3_{NoAttack}$, $XMEAS4_{NoAttack}$,$XMEAS5_{REPLAY(70)}$,$XMEAS7_{DOS(70)}$, $XMEAS8_{REPLAY(30)}$,$XMEAS9_{DOS(70)}$,$XMEAS10_{REPLAY(70)}$, $XMEAS11_{NoAttack}$,$XMEAS12_{NoAttack}$,$XMEAS14_{NoAttack}$, $XMEAS15_{REPLAY(70)}$,$XMEAS17_{NoAttack}$,$XMEAS31_{NoAttack}$, $XMEAS40_{REPLAY(35)}$,$XMV1_{NoAttack}$,$XMV2_{DOS(35)}$, $XMV3_{NoAttack}$,$XMV4_{DOS(15)}$,$XMV6_{DOS(65)}$, $XMV7_{NoAttack}$, $XMV8_{NoAttack}$, $XMV10_{NoAttack}$, $XMV11_{REPLAY(70)}$
Damage ($)	170.00
Detection probability	28.0

Table 8.2: Description of an individual (attack) generated by the EMO

Selection Operator($)	$(\mu+\lambda)$ Strategy	(μ,λ) Strategy ($)
SPEA2	457	83
NSGA-II	352	74
Average Hypervolume	0.6517	0.2739

Table 8.3: Results for $\mu+\lambda$ and μ,λ strategies

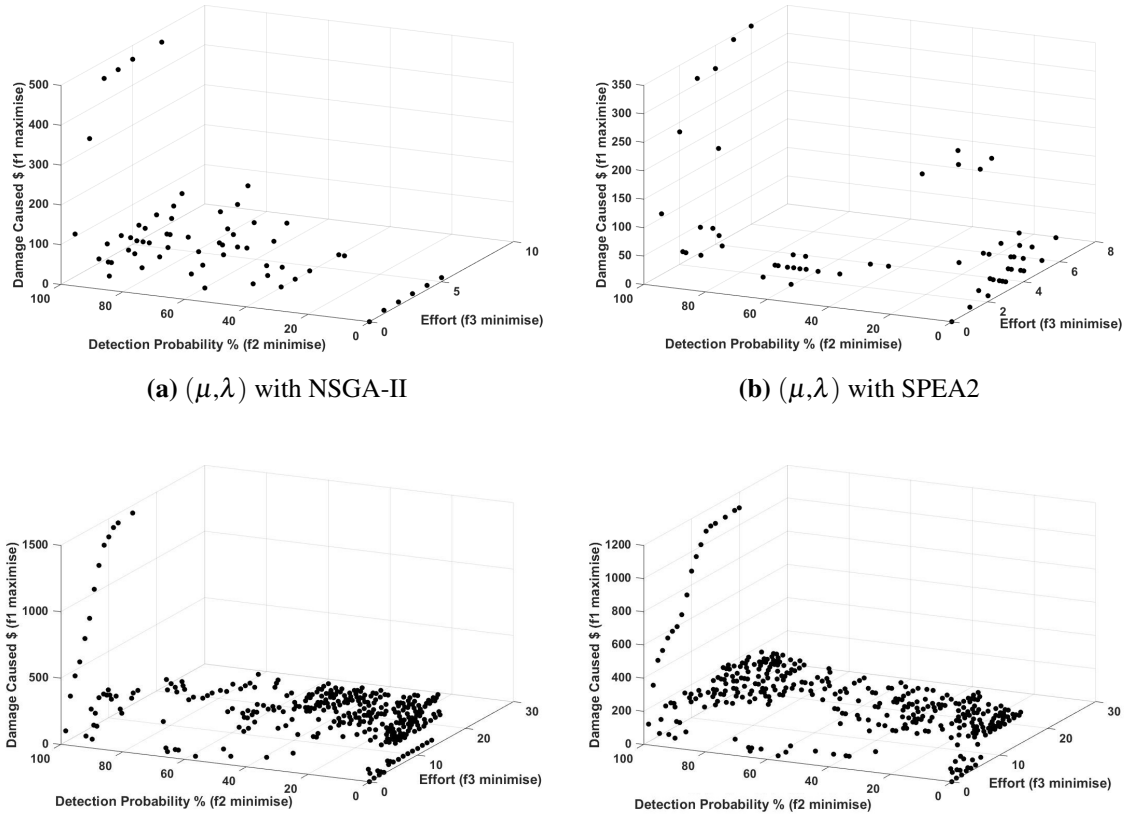
resource and time limitations precluded this. As indicated in Table 8.3, the number of elements in the Pareto front set, and the quality of the elements in the set as indicated by the hypervolume were significantly better for the $(\mu+\lambda)$ strategy. Based on the these results, the $(\mu+\lambda)$ strategy was used to carry out further experiments to generate attacks against the detection models.

The results averaged over two runs using the $(\mu+\lambda)$ strategy are illustrated in Table 8.4.

Performance Measure	NSGA-II	SPEA2
Unique Attacks	118,006	100,387
Damage Range ($)	0-1633	0-1114
Cardinality of Pareto Front	330	412
HyperVolume	0.7416	0.6653

Table 8.4: Results for attacks generated against AdaBoost (averaged over all runs)

NSGA-II focuses on maximising spreading, and this is reflected in the re-



(a) (μ,λ) with NSGA-II

(b) (μ,λ) with SPEA2

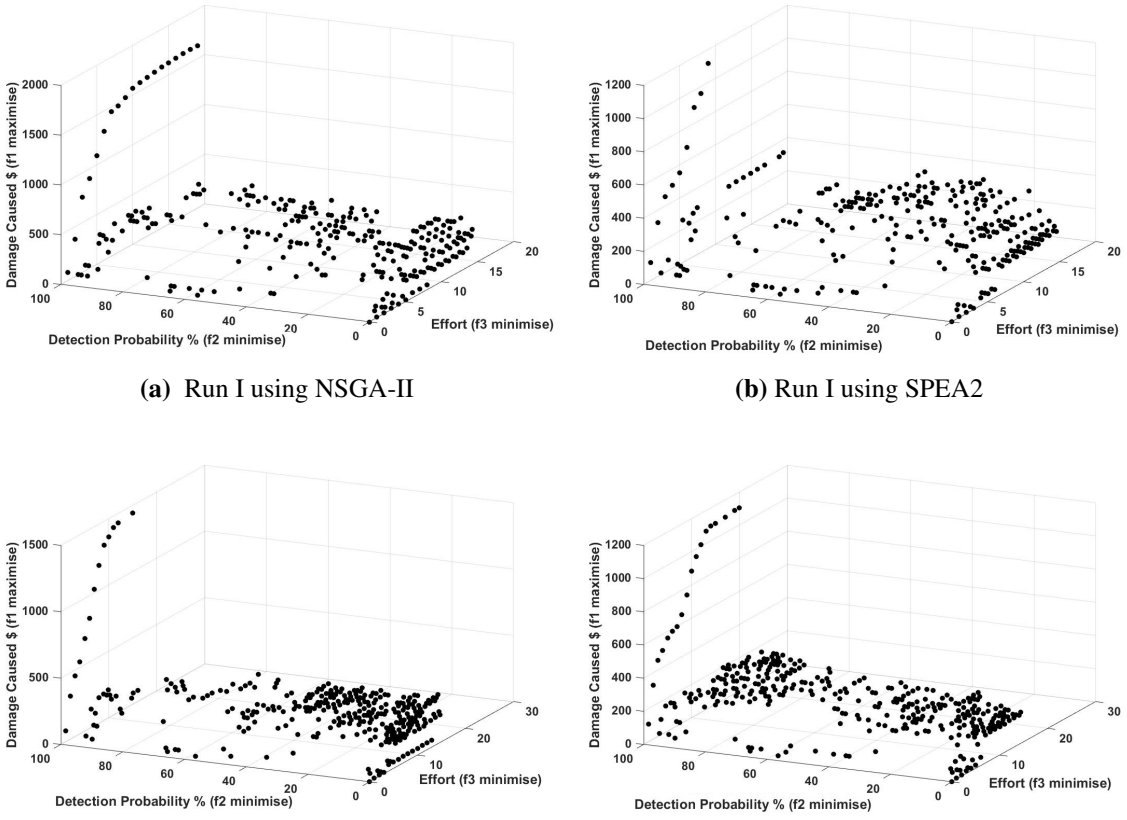
(c) $(\mu+\lambda)$ with NSGA-II

(d) $(\mu+\lambda)$ with SPEA2

Figure 8.3: Pareto fronts of attacks generated using (μ,λ) and $(\mu+\lambda)$ strategies

sults, giving it a better spread of attack, and hence a better value for hypervolume. Throughout the process of evolution, NSGA-II generated, on average, 118,006 unique individuals with a range of damage from $0-1633. SPEA2 generated 100,387 unique individuals with a range of $0-1144. Although, the cardinality of the Pareto front for the final population was larger for SPEA2; on average there were 412 elements in the Pareto front set compared to 330 in the sets generated using NSGA-II, the attacks were less optimal than those found using NSGA-II. However, the distribution of the individuals appears to be better for SPEA2, as illustrated in Figure 8.4b and Figure 8.4d.

More experiments are required for a reliable comparison; however, overall, both algorithms found attacks that avoid detection and cause some harm. Assuming the detection probability threshold is less than 5%, the highest cost attack NSGA-II



(a) Run I using NSGA-II

(b) Run I using SPEA2

(c) Run II using NSGA-II

(d) Run II using SPEA2

Figure 8.4: Pareto fronts of two best runs: attacks generated against AdaBoost using $(\mu+\lambda)$ strategy (3-Objective optimisation)

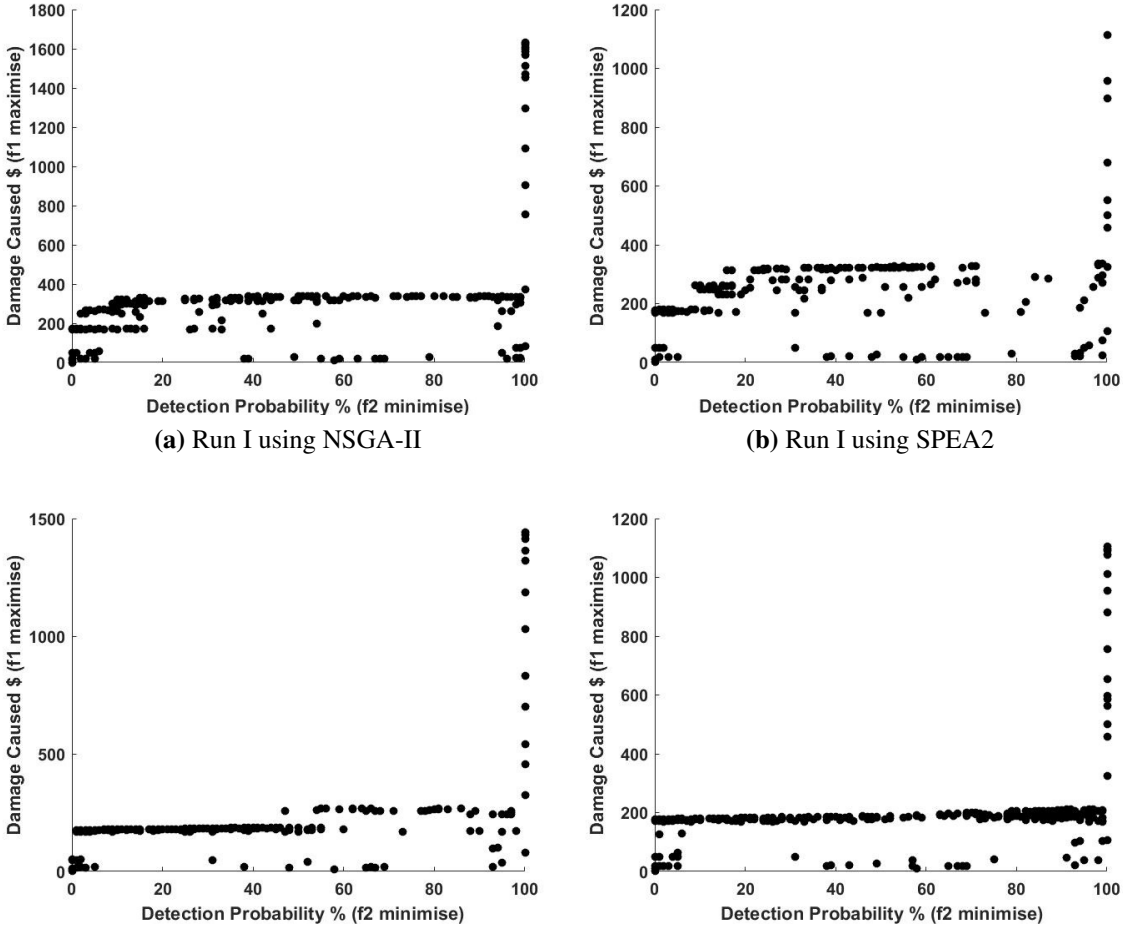
found is $266.92, attacking 12 sensors and actuators, with detection probability 3%, whereas SPEA2 found a slightly worse attack, $179.72 with effort 16 and detection probability 4%. If the intention is to use the smallest effort and cause maximum damage, then the optimal attack strategy produced using NSGA-II is an attack that costs $179.72 using effort 6. SPEA2 found a similar attack using effort 6, at a cost of $170.69. The descriptions of these attacks are given in Table 8.5. The attacks generated using less effort were either detected or caused low economic damage, that is $\leq$ $50.00.

Plotting the 2-objectives of the Pareto front set against each other (the damage caused by detection probability), shows an interesting pattern. As illustrated in Figure 8.5, as the damage caused against attacks gets significantly higher, the detection probability also increases. For attacks ranging around $100-250, where some

Attack generated using NSGA-II	Damage ($)	Detection (%)	Effort
$XMEAS2_{REPLAY(70)}$,$XMEAS5_{REPLAY(70)}$, $XMEAS10_{REPLAY(70)}$,$XMEAS11_{REPLAY(70)}$, $XMEAS12_{DOS(65)}$,$XMEAS15_{REPLAY(70)}$, $XMEAS17_{REPLAY(70)}$,$XMEAS31_{DOS(70)}$, $XMEAS40_{REPLAY(3)}$,$XMV4_{REPLAY(65)}$, $XMV10_{REPLAY(70)}$,$XMV11_{REPLAY(70)}$	266.92	3.0	12.0
$XMEAS3_{REPLAY(70)}$,$XMEAS5_{REPLAY(70)}$, $XMEAS7_{DOS(70)}$,$XMEAS10_{REPLAY(70)}$, $XMV2_{REPLAY(65)}$,$XMV11_{REPLAY(70)}$	170.00	4.0	6.0
Attack generated using SPEA2	Damage ($)	Detection (%)	Effort
$XMEAS1_{DOS(63)}$,$XMEAS2_{REPLAY(70)}$, $XMEAS3_{REPLAY(70)}$,$XMEAS4_{DOS(63)}$, $XMEAS5_{DOS(70)}$,$XMEAS7_{DOS(70)}$, $XMEAS8_{DOS(20)}$,$XMEAS10_{REPLAY(70)}$, $XMEAS15_{REPLAY(70)}$,$XMEAS17_{REPLAY(50)}$, $XMEAS31_{DOS(70)}$,$XMEAS40_{DOS(27)}$, $XMV2_{REPLAY(65)}$,$XMV4_{REPLAY(63)}$, $XMV6_{DOS(6)}$,$XMV11_{REPLAY(70)}$	179.73	4.0	16.0
$XMEAS5_{REPLAY(70)}$,$XMEAS7_{DOS(70)}$, $XMEAS8_{REPLAY(70)}$,$XMEAS9_{DOS(70)}$, $XMEAS10_{REPLAY(70)}$,$XMEAS40_{DOS(27)}$	170.69	4.0	6.0

Table 8.5: Examples of high scored attacks generated using NSGA-II and SPEA2

of the best individuals were found in terms of the trade-off, both algorithms try to maintain more individuals around this area. As compared to the 2-objective optimisation problem discussed in Section 8.2.1, the 3-objective optimisation requires us to optimise the effort. The results indicate that attacking a higher number of sensors and actuators may not always lead to a higher detection probability. That is, if the attacker attacks the correct combinations of the sensors and actuators, it can ensure that the plant operates closer to the normal operating conditions and, therefore, one can evade detection. This is further illustrated in Table 8.6.



(a) Run I using NSGA-II

(b) Run I using SPEA2

(c) Run II using NSGA-II

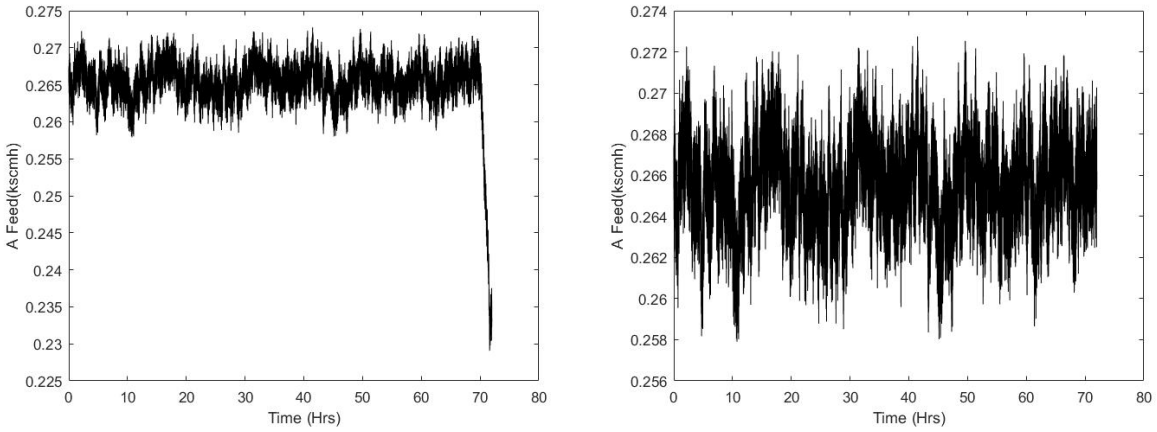
(d) Run II using SPEA2

Figure 8.5: Comparing two objectives of the Pareto front: damage caused against detection probability

Attack	Damage ($)	Detection (%)	Effort
$XMEAS1_{DOS(65)}$,$XMEAS2_{DOS(27)}$, $XMEAS3_{DOS(56)}$,$XMEAS4_{DOS(65)}$, $XMEAS7_{REPLAY(70)}$,$XMEAS12_{REPLAY(33)}$, $XMEAS17_{DOS(60)}$,$XMEAS31_{DOS(63)}$, $XMV10_{REPLAY(35)}$,$XMV11_{REPLAY(65)}$	1567.43	100	10
$XMEAS2_{DOS(70)}$,$XMEAS5_{REPLAY(70)}$, $XMEAS8_{REPLAY(70)}$,$XMEAS9_{REPLAY(70)}$, $XMEAS10_{REPLAY(70)}$,$XMEAS12_{DOS(65)}$, $XMEAS15_{REPLAY(70)}$,$XMEAS17_{REPLAY(70)}$, $XMEAS31_{DOS(70)}$, $XMV2_{REPLAY(60)}$	314.44	14	10
$XMEAS2_{DOS(70)}$,$XMEAS5_{REPLAY(70)}$, $XMEAS9_{REPLAY(70)}$,$XMEAS10_{DOS(70)}$, $XMEAS15_{REPLAY(70)}$	304.24	99	5
$XMEAS5_{REPLAY(70)}$,$XMEAS7_{DOS(70)}$, $XMEAS10_{REPLAY(70)}$,$XMEAS12_{REPLAY(70)}$, $XMEAS14_{REPLAY(65)}$,$XMEAS15_{REPLAY(70)}$, $XMEAS31_{DOS(70)}$,$XMEAS40_{DOS(27)}$, $XMV4_{REPLAY(65)}$,$XMV11_{REPLAY(70)}$	171.12	1	10
$XMEAS5_{REPLAY(70)}$,$XMEAS7_{DOS(70)}$, $XMEAS10_{REPLAY(70)}$,$XMEAS12_{REPLAY(70)}$, $XMV2_{REPLAY(65)}$	170.45	44	5
$XMEAS5_{REPLAY(70)}$,$XMEAS7_{DOS(70)}$, $XMEAS10_{REPLAY(70)}$,$XMEAS12_{REPLAY(70)}$, $XMEAS15_{REPLAY(70)}$,$XMEAS31_{DOS(70)}$, $XMV2_{REPLAY(60)}$,$XMV4_{REPLAY(65)}$, $XMV11_{REPLAY(70)}$	167.92	0	9
$XMEAS9_{REPLAY(70)}$	81.10	100	1
$XMEAS5_{REPLAY(70)}$,$XMEAS7_{DOS(70)}$, $XMEAS10_{DOS(70)}$,$XMEAS12_{REPLAY(70)}$, $XMEAS31_{DOS(65)}$,$XMEAS40_{REPLAY(3)}$	50.93	0	6

Table 8.6: Some of the individuals in the Pareto front generated using NSGA-II

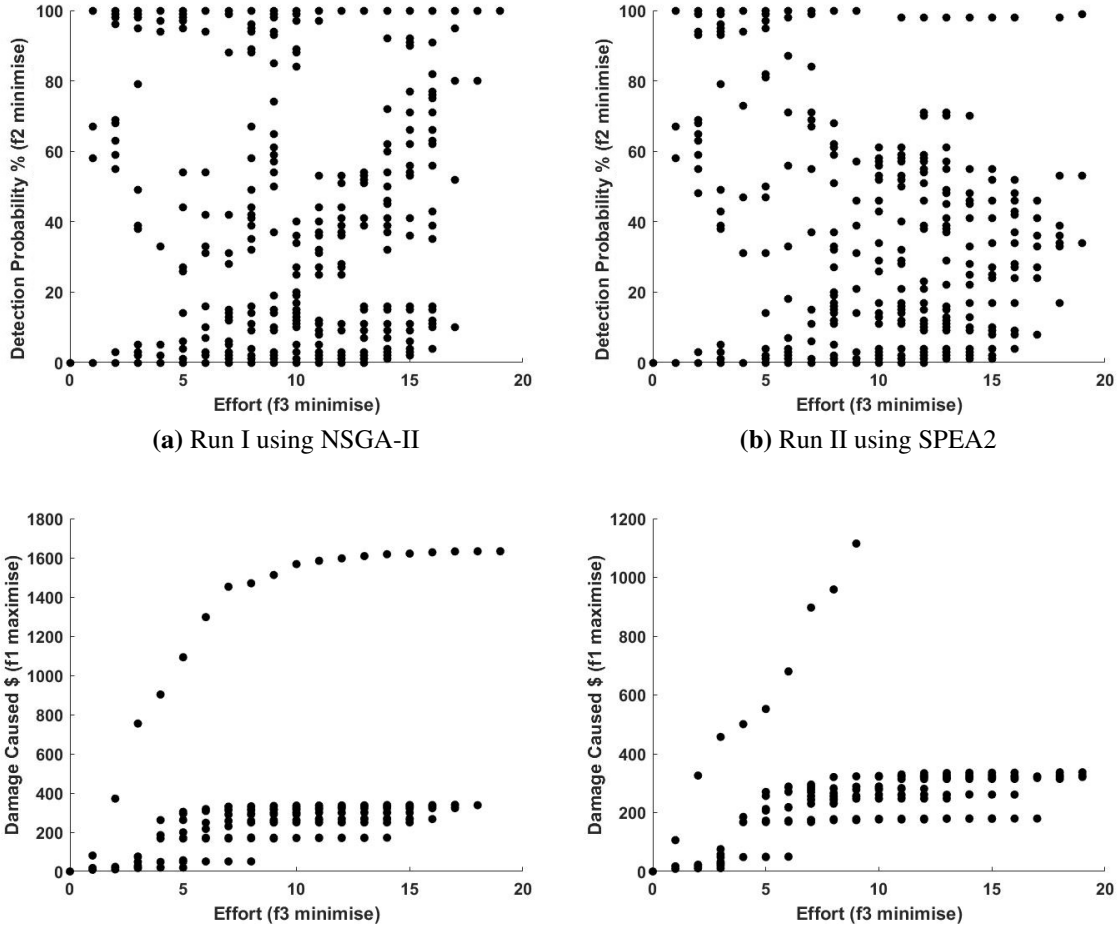
The EMO was able to find the right combinations of sensors and actuators to evade detection while causing some economic damage, and this may not involve attacking fewer sensors and actuators, seen in the fourth row. Carrying out a single replay attack (Attack I) against the XMEAS 9 (Reactor Temperature) is detected immediately with a detection probability of 100%, whereas combining these attacks with others, as seen in the second row (Attack II), reduces the attack detection probability to 14%. The impact of these two attacks on the A Feed flow (XMEAS 1) is shown in Figure 8.6. The single replay attack, Attack I, causes the A Feed to go below the normal range (0.25-0.27 kscmh), but the combined attack Attack II, is able to keep the A Feed around the normal range. Most of the attacks generated by our EMO that evade detection while causing some damage are generated around the shutdown time of the plant, at hour 70, and these attacks last for the two hours immediately prior to the completion of production. It is likely that the plant had no time to react to the consequences of these attacks and our EMO was able to exploit this weakness.



(a) Attack I Detected **(b)** Attack II Not Detected

Figure 8.6: Impact of the attack on A Feed (XMEAS I)

Figure 8.7a and Figure 8.7b are plots that show effort against the detection probability for the attacks in the Pareto front for Run I. Other runs had similar results. Each attack lasts for 2 hours and using more effort does increase the damage, as shown in Figure 8.7c and 8.7d; however, these high cost attacks also increase the



(a) Run I using NSGA-II

(b) Run II using SPEA2

(c) Run I using NSGA-II

(d) Run II using SPEA2

Figure 8.7: Comparing objectives: effort against detection probability and damage against effort

probability of detection. The optimisation finds the right balance, by carrying out attacks that cause less damage but evade detection, indicated by the increased spread of points along the lower impact attacks that cause economic harm of $\leq$ \$300.

Figure 8.8 shows the hypervolume indicator for the selection methods, NSGA-II and SPEA2, using the selection strategy (μ,λ), averaged over Runs I and II for 800 generations. As before, the values are normalised so that they lie between 0 and 1. Both optimisations started from the same initial population, which was changed between Run I and Run II. SPEA2 appears to perform better than NSGA-II at the beginning of the evolution, and leads until generation 158, after which it starts to be overtaken by NSGA-II, with both reach similar hypervolume around generation

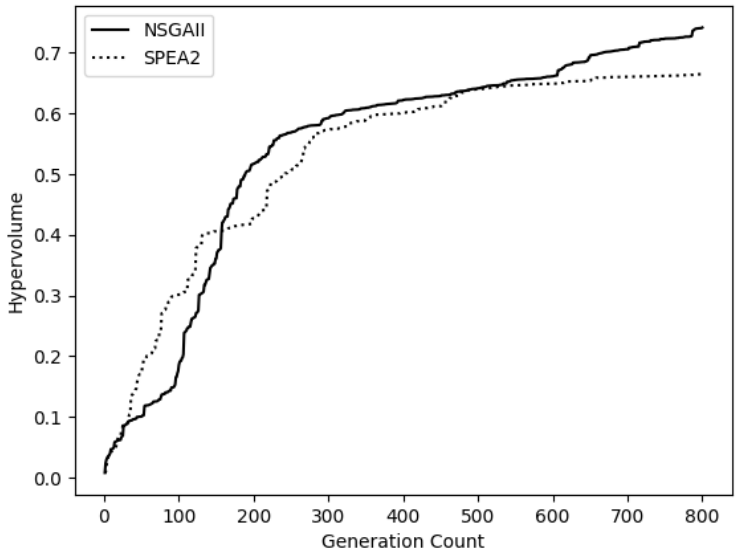


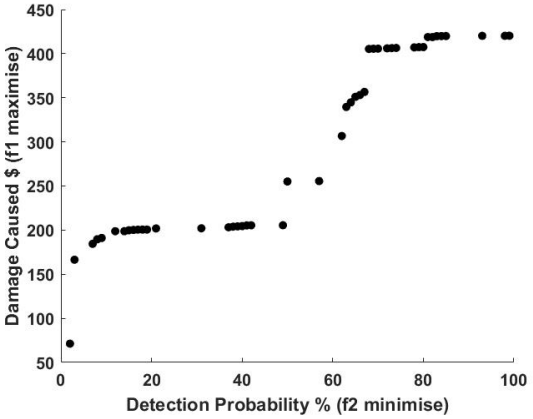
Figure 8.8: Hypervolume results for NSGA-II and SPEA2 (averaged and normalised over all runs)

480. By generation 550, NSGA-II overtakes SPEA2 again. Overall, for these runs NSGA-II shows better performance, finishing with a hypervolume of 0.74 in comparison to SPEA2 at 0.66. This is due to NSGA-II having a better spread for this problem, having some individuals able to cause more damage than the individuals obtained from SPEA2. However, those attacks that cause damage and evade detection for both algorithms indicate a similar performance. The experiments were executed for 800 generations but it is clear from Figure 8.8 that both algorithms require more time to converge properly; this is especially the case for NSGA-II as it has been steadily increasing.

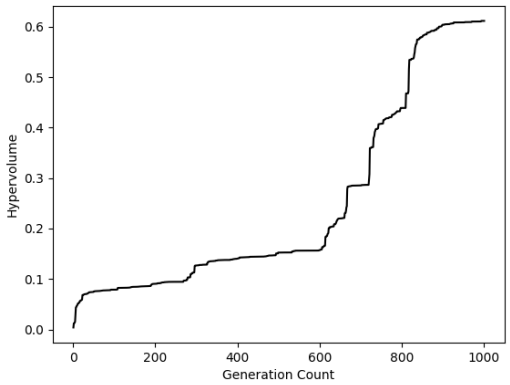
8.2.3 Generating Attacks against Decision Tree and Random Forest

We did not have enough time and resources to test our EMO comprehensively against all supervised classifiers; however, we carried out three sets of runs against the single Decision Tree and Random Forest classifiers, and both performed better than the AdaBoost classifier on DoS and replay attacks, albeit with at a higher rate for false positives. For these experiments, the genetic parameters in Table 8.1

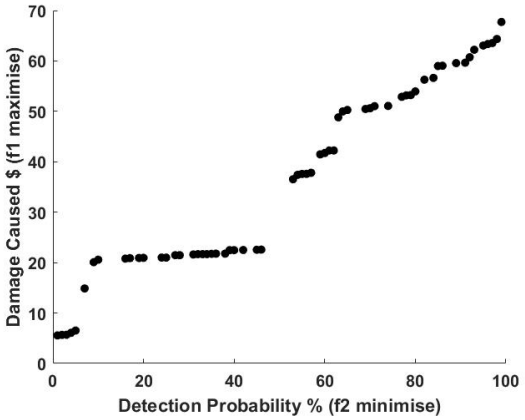
were used in combination with the NSGA-II selection method and μ+λ selection strategy. The duration of the attacks were a constant 2 hours.



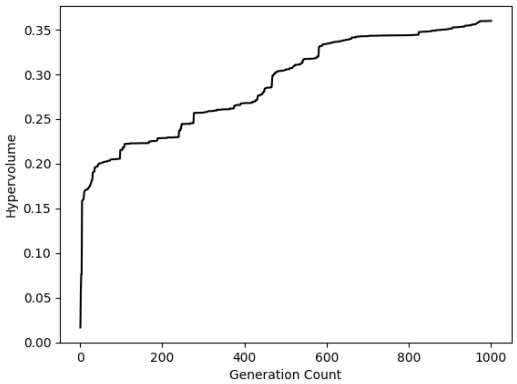
(a) Pareto front of attacks generated against Decision Tree



(b) Hypervolume of Decision Tree



(c) Parato front of attacks generated against Random Forest



(d) Hypervolume of Random Forest

Figure 8.9: Generated attacks against Decision Tree and Random Forest (2-Objective optimisation)

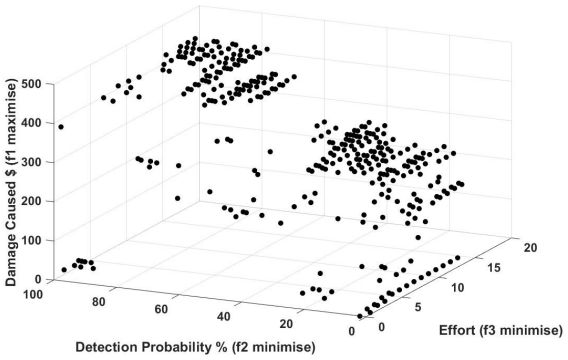
Figure 8.9 shows the best run for the Decision Tree and Random Forest. Decision Tree was able to find a wide variety of attacks that caused damage, including an attack that caused damage of $166 with a detection probability of 3%, as shown in Figure 8.9a. Given the false positive rates of Decision Tree, one needs to select a threshold for detection probability higher than the AdaBoost classifier. Assuming a threshold of 60% is chosen for declaring an alarm, as shown in Figure 8.9a we found some attacks that increase the operating cost by $255 that the intrusion

system will miss. The increasing hypervolume convergence curve shown in Figure 8.9b suggests that running the EMO for more generations could have produced better attacks.

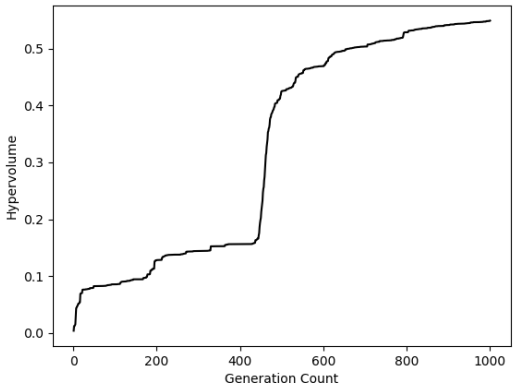
Random Forest had a better detection probability for DoS attacks than the Decision Tree classifier and One-Class SVM, thus, it is harder to generate attacks against this classifier. Figure 8.9c shows the Pareto front of the best run we obtained from experiments carried out with the Random Forest classifier. Pareto front set has attacks that increased the operating cost between \$5 and \$67. As before, assuming a threshold of 60% is chosen for declaring an alarm, there are some attacks that increase the operating cost by \$41 that the intrusion system will miss, as shown in Figure 8.9c. Figure 8.9d shows the convergence curve is steadily increasing, and there is potential for generating attacks that cause more damage while evading detection. As with the Decision Tree, these results also show that running the EMO for more than 1000 generations could generate a better Pareto front and achieve a asymptotic convergence.

We also carried out three sets of experiments with three objectives (maximise damage cost, minimise detection probability, minimise effort) against the Decision Tree and Random Forest classifiers using the genetic parameters in Table 8.1 in combination with the NSGA-II selection method and μ+λ selection strategy, to search for attacks that cost the least effort. Figure 8.10 shows the best results obtained obtained over 1000 generations.

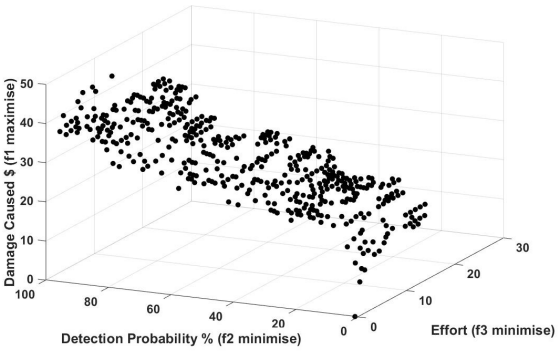
EMO was successful at finding a range of attacks that evade detection using a range of efforts. These results are similar to those obtained against the 2-objectives (Figure 8.9). As before, EMO was able to make better use of the weaknesses of the Decision Tree classifier, and was able to find attacks that evade detection with probability of detection $< 60\%$, while causing economic harm of $\leq$ \$350. As shown Figure 8.10a the effort required to carry out these attacks is often high. Although, EMO was also able to also find some attacks that cause economic damage of $\$186 - 152$ using a range of efforts $6-4$, it failed to find attacks that cause a significant damage using efforts < 4. The increasing hypervolume convergence curve shown in Figure



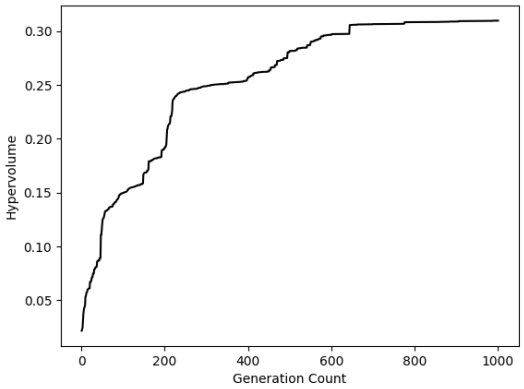
(a) Pareto front of attacks generated against Decision Tree



(b) Hypervolume of single Decision Tree classifier



(c) Parato front of attacks generated against Random Forest

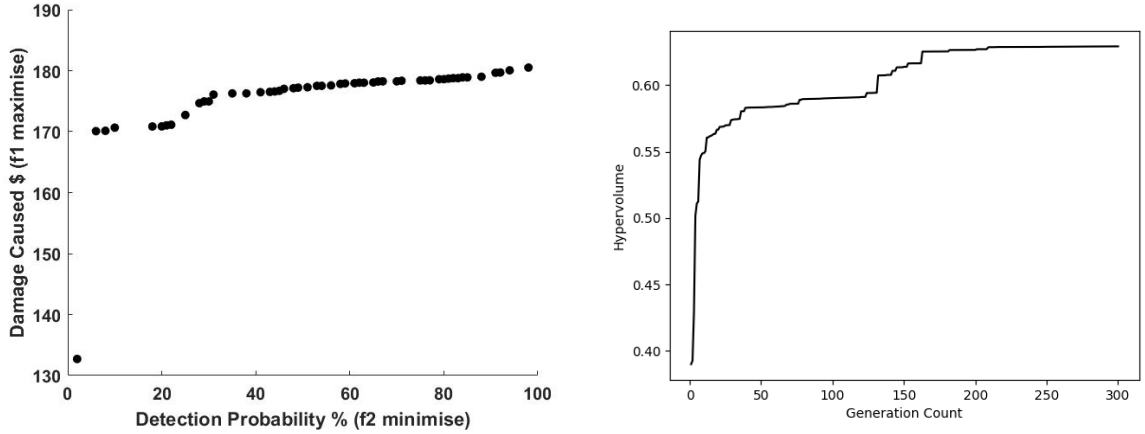


(d) Hypervolume of Random Forest classifier

Figure 8.10: Generated attacks against Decision Tree and Random Forest (3-objective optimisation)

8.10b shows, once again, we need to run the EMO for more generations, and we may evolve better attacks. For Random Forest most of the evolved attacks caused a small increase in the operating cost of the plant, just under 30, and as shown in Figure 8.10c, there are no significant changes after generation 650 indicating a possibility of a premature convergence, as shown in Figure 8.10d.

To test if we could generate better attacks against the Random Forest classifier, we carried out an experiment in which we added some good individuals that we had obtained previously from the experiments carried out against the Decision Tree classifier, to the initial random population. This is known as seeding the popula-



(a) Pareto front of attacks generated from a seeded population

(b) Hypervolume of attacks generated from a seeded population

Figure 8.11: Generated attacks from a seeded population against Random Forest

tion and uses a few candidate individuals that have previously been determined to be good (e.g. using expert knowledge) or that have been generated as a result of some problem solving approach. Seeding is a common practice in single-objective evolutionary algorithms; however, the benefits and disadvantages of seeding in evolutionary multiobjective algorithms are not well studied for real-world problems. In particular, the few of the existing studies show it can reduce the computational cost especially when fitness function evaluation is expensive, and thus improve the speed of the search, and quality of the solutions [308]. A comprehensive study of this research gap is future work, however, we carried out an experiment where we seeded the initial population with 10 individuals that the Decision Tree failed to detect, and started the EMO from this modified population against the Random Forest classifier. Figure 8.11a shows the result of our experiments after 300 generations.

Results shows the evolved attacks are not better than those against the Decision Tree, and this was expected since Random Forest had a better detection performance than Decision Tree, but, these attacks are also significantly better than those attacks that started from a completely random population. The Pareto front includes attacks that increase the operating cost by $170 with a detection probability less than 10%. Our experiment shows that individuals obtained from a weak classifier can be used to seed a strong classifier and so to reduce the computational effort. However,

seeding can also cause the population to lose diversity rapidly, and decrease the explorative ability of the EMO, causing the population to become trapped in local optima. Figure 8.11b illustrates the possibility of this problem, as the hypervolume begins from 0.40 and increases to 0.62, and there is no significant change after generation 162. These results indicate that the seeded good individuals have taken over the population, and evolved attacks are derived predominantly from those initial seeds. It is thus important to find a balance to ensure that randomly generated individuals are effectively combined with the seeds produce attacks.

8.2.4 Generating Attacks against One-Class SVM

Using the EMO we evolved attacks against the unsupervised classifier, One-Class SVM. The F1 scores for One-Class SVM were 57.01% for single DoS attacks and 47.61% for single replay attacks. The false positive rate for One-Class SVM was 0.44%, higher than the decision tree based classifiers.

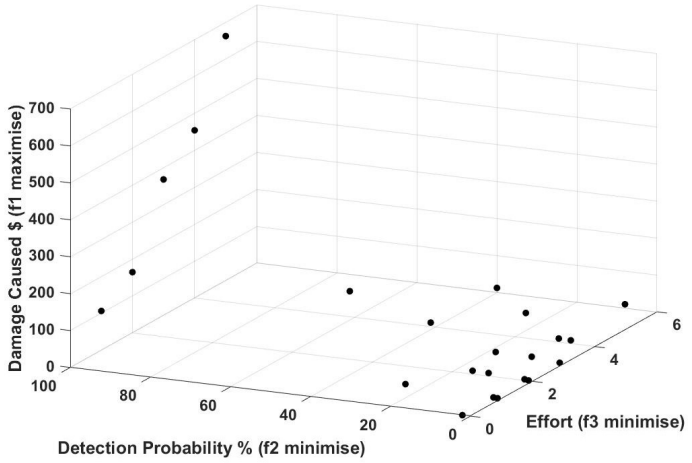
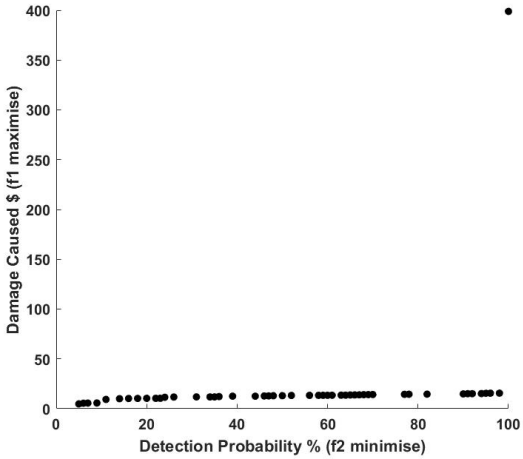
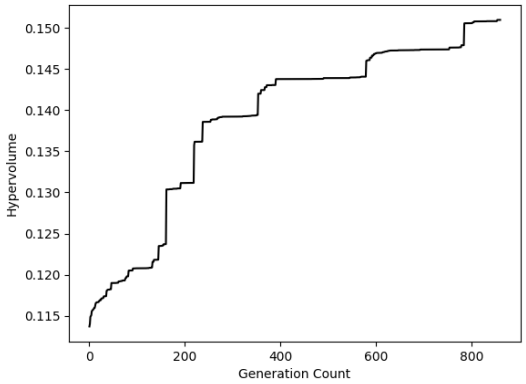


Figure 8.12: Generated DoS and replay attacks against One-Class SVM (3-Objective optimisation)

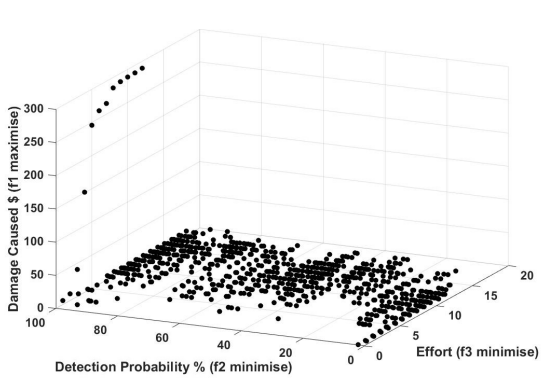
Using the same genetic operators and parameters as in the previous experiments, and limiting the number of attacks for each chromosome in the initial population to fewer than 4, we carried out replay and DoS attacks against the One-Class SVM: the duration of the attacks was again a constant 2 hours using the EMO with the NSGA-II selection method. Figure 8.12 shows the results obtained from one of



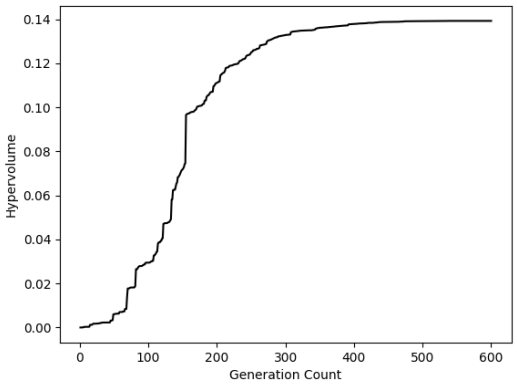
(a) Pareto front of DoS and replay attacks generated using a seeded population (2-Objective optimisation)



(b) Hypervolume of DoS and replay attacks generated using a seeded population



(c) Pareto front of replay attacks (3-Objective optimisation)



(d) Hypervolume of replay attacks

Figure 8.13: Generated attacks against One-Class SVM using seeding and evolving only replay attacks

these experiments. One-Class SVM was able to detect these attacks; the maximum damage that can be inflicted on the plant without getting detected was just $12, with a detection probability of 40% and a need to attack 5 sensors and actuators. Other runs showed similar performance and failed to find any attacks causing significant damage.

We used seeding to see if we can generate better attacks against One-Class SVM by including some of the best individuals that were obtained from the Decision Tree classifier to the initial population, and evolved attacks against the 2-objective (maximise damage, minimise detection probability. Figure 8.13a illus-

trates the obtained Pareto front. Seeding the initial population helped to evolve attacks faster than the random population and found more attacks against the SVM, but, the damage inflicted was not significantly better: it increased the operating cost by $13.5 with a detection probability less than 60%. Figure 8.13b illustrates the hypervolume is increasingly slowly, and there might be some space for generating better attacks, however it is unlikely to find attacks that increase the operating cost significantly while evading detection.

To give EMO some space to work, we carried out some experiments in which we: increased the attack detection window size to 300 instead of 100 as used in the previous experiments; reduced the duration of the attacks to between 30 minutes and 1 hour; and used only replay attacks, as DoS attacks were easier to detect. Initial population was generated randomly. The start time of the attack was any hour from 2-70. This range was used to define the type of gene. As before, the experiments were carried out using utilising the selection method of NSGA-II for the 3-objective (maximise damage, minimise detection probability, minimise effort) case. Figure 8.13c illustrates Pareto front of onen of the best runs. Compared to decision tree classifiers, the performance of the One-Class SVM is slower, and, therefore, we are limited to a smaller generations. The 3-objective EMO found a variety of attacks that evade detection, but the damage these attacks caused was under $10. Figure 8.13d, the evolution converged and there is apparently little space for further improvement. Overall, these results indicates that One-Class SVM is able to detect attacks better than the decision tree classifiers, but at a higher cost of false positive rate. Thus, the EMO had a very little space to evolve attacks that could cause significant damage using the designed attacks.

8.3 Application of the EMO Approach against other Industrial Control Systems

The evolutionary multiobjective optimisation approach developed in Chapter 6 and this chapter can be employed to test the vulnerabilities of systems with a broad attack surface including ICS domains such as: electricity, oil and gas production

and distribution; water and waste treatment systems; space; and manufacturing (e.g. metal-alloy manufacture, pharmaceuticals). Attacks in such domains target sensor measurements that are sent to the controllers and manipulated values that are sent from controllers to the actuators, as these values can impact the process directly; and achieve the intended damage.

The approach described in this thesis is agnostic to the application domain: it can be applied to other process control networks with a high number of sensors and actuators. To successfully apply it to other systems, one needs to define the objectives and the search parameters. Search parameters include defining the potential targets (sensors and actuators), and attack parameters including start time, duration and attack pattern (e.g. periodic and interval). For man-in-the middle attacks, it is necessary to establish the lower and upper range of process variables and manipulated variables. In this thesis, we employed a small range of values for start time and attack duration as we were limited by the evaluation speed of the fitness function; however, given sufficient resources a large range can be used. Expert knowledge can be used to tune the search parameters to save time. For example, those processes that exhibit faster dynamics, as opposed to low dynamic processes like the TE process, will cope better with attacks that are shorter in duration. In generating attacks, we treated detection as a black-box; the only information EMO requires from the detection is the detection probability. After this, EMO can be left to search for attacks.

8.4 Summary

In this chapter, we investigated Research Question 4: Can one evolve new attacks against the Intrusion Detection System? We evolved attacks against the detection methods implemented in Chapter 7: AdaBoost, Decision Tree and Random Forest and One-Class SVM.

We were able to generate a large number of attacks that were not detected by all detection systems. However, generating attacks that evade detection, and at the same time cause some significant economic damage on a system like TE process is

a challenging task, since it requires attacks to be long in duration. Our results show that carrying out a successful attack requires knowledge of the system to ensure that the correct combination of sensors and actuators are attacked, and to avoid detection or the triggering of the safety system. The significant attacks generated against the AdaBoost, Decision Tree and Random Forest classifiers involved attacking multiple components in the system to evade detection while causing economic damage. A naive attacker that randomly attacks multiple targets is unlikely to achieve similar damage, and is highly likely to be detected. The focus of our future work will involve investigating and designing more complex attacks that could learn the behaviour of the plant and the detection system, and generate attacks using this knowledge. Despite this, the results obtained show that evolutionary multiobjective optimisation can be used successfully as an effective tool to model the behaviour of the adversary against a defence mechanism, and illustrate the conflicts and associated trade-offs among common security objectives: attack impact, detection and effort required. Using the results obtained from such an analysis, security engineers can take measures to understand and eliminate system vulnerabilities before they are exploited by malicious actors.

Next we summarise some issues related to the implementation of our approach. Formulating the problem using multiple objectives increases its complexity. It is often possible that the optimisation methods that work well for 2 objective problems may not perform equally well for a higher dimensional objective space as the number of non-dominated individuals increases. Utilising the weakness of the AdaBoost classifier, we were able to carry out experiments using a large space to utilise and compare selection methods of two well-known evolutionary optimisation algorithms. Using the $(\mu+\lambda)$ strategy, both NSGA-II and SPEA2 show promising performance; however the results indicate that NSGA-II appears to search the space better when using 3 objectives, maximising the spread of the solutions. Generating attacks against a strong classifier can be a very time consuming task; this is especially true for cases like our model, for which the evaluation of individuals (i.e. the fitness function) is slow. The slowness of our fitness function also influenced

the way we encoded our individuals, the size of the population, and the number of generations.

Incorporating some knowledge into the EMO, as we have done with Random Forest by seeding the population with some individuals obtained from a weaker classifier (single Decision Tree), shows that it can significantly reduce the duration of experiments and more rapidly identify those attacks that are likely to evade detection. However, a comprehensive study is required to understand the full benefits and weaknesses of the variety of strategies for this approach (e.g. such as the number of seeds used), as seeding can reduce the diversity of the population and, so fail to explore other feasible region in the attack space. In future work, we plan to investigate effects of seeding and possible seeding strategies as there is little work in this area applied to real-world multiobjective combinatorial optimisation problems.

Chapter 9

Conclusions and Future work

To conclude, in this chapter we summarise our findings, key contributions and potential directions for future work.

9.1 Summary

Although the security of industrial control systems is attracting more research, the nature of vulnerabilities and the attacks that can be carried out as a consequence of exploiting these vulnerabilities are not well understood at present. Studying this can help to identify weaknesses in the process design of these systems, and weaknesses of any existing detection systems in place to protect these systems. These systems are composed of a large number of sensors, actuators and controllers. Carrying out attacks against these systems is a complex task due to size of the possible set of targets and attack types. Efficient and effective tools that can automate this process can help to examine these systems against attacks from a wide variety of adversaries with different motivations. In this thesis, we investigated an approach to achieve precisely this.

In our approach, we studied and used a well-known benchmark chemical process, the Tennessee Eastman process control problem, and extended an implementation to study the attacks that could be carried out by potential adversaries. We first studied the impact of single attack instances on sensors and actuators using DoS, integrity and replay attacks where the effect of the attack was measured in terms of safety and economic loss. We assumed that adversaries might have a goal to attack

the safety of the system by forcing the process into an unsafe state, causing it to violate the shutdown limits and therefore shutting it down. Alternatively, we assumed that some adversaries that may wish to cause damage by increasing the operating cost of the process.

To study more complex attacks involving combinations of attack instances, we designed experiments to compare random search, genetic algorithms with a single objective, and genetic algorithms with multiobjective optimisation. To study the effectiveness of genetic algorithms, we carried out a set of experiments to compare the performance of a simple genetic algorithm with random search when evolving DoS attacks. The genetic algorithm outperformed random search by generating attacks that were much better in fitness and distribution. This single objective genetic algorithm was successful at finding combinations of attacks that caused more damage than single attacks; however, it demonstrated one of the key weaknesses of the approach. The attacks were not necessarily minimal, because it was not possible to know if the same attack could have been carried out by attacking a smaller target (i.e. fewer sensors and actuators).

As a result, the problem was reformulated as an evolutionary multiobjective optimisation problem using the selection method of two respected optimisation algorithms, NSGA-II and SPEA2. The performance of these algorithms was evaluated using the hypervolume metric to measure the quality of the Pareto set. The performance of both algorithms was similar for this task: they found overlapping sets of attack strategies. However, SPEA2 had a slightly better performance, and it found the Pareto set faster than NSGA-II. Although, in some cases, premature convergence can occur, overall our experiments showed that genetic algorithms with multiobjective optimisation can be used effectively to determine vulnerabilities in the process, by analysing the consequences of the attacks generated. We categorised some of the most vulnerable combinations of sensors and networks, and recommend these combinations should be considered carefully when designing process control strategies and cost-effective network security hardening.

To detect these attacks, a set of classic machine learning algorithms and deep

learning algorithms were used to implement intrusion detection mechanisms. Supervised learning methods for attack identification require attack examples, and the process of labelling this data accurately may not be easy as one needs to determine when the process returns to normal. This is not a task that can be reliably carried out by a human. We adopted an approach that relies on a modified Z-score to estimate when the process returns back to normal. It is highly possible that we might have misclassified some of the data points, although this is likely unimportant in a classifier used to noisy input data. Although some more work could have been done to improve the performance of these classifiers by searching a large number of parameters to ensure the optimal hyperparameters were used, the performance is unlikely to have changed significantly. The results show that, as is often the case with supervised learning, their performance reduces significantly on unseen cases. The supervised methods did well on detecting integrity attacks but performed less well on DoS and replay attacks. On the other hand, unsupervised learning methods did better overall, but the detection was noisy, yielding a higher false positive rate. To handle the false positives, a sliding window was used for all detection methods; the size of the sliding window depends on the patterns of the false positives. Two forms of recurrent neural network, LSTM and GRU, had a higher false positive rate than One-Class SVM but the windows of consecutive false positive data points were much smaller. This was beneficial both for accuracy declaring in attacks and for early detection. Overall, the autoencoder outperformed all unsupervised approaches. However, in general, deep learning methods are harder to train in comparison to more conventional algorithms like One-Class SVM since they have large number of hyperparameters and are sensitive to the choice of these parameters.

After defining a set of methods for detection, the final part of this PhD was to evolve attacks against the improved defence, again searching for attacks that could cause economic loss and, at the same time, evade detection. Attacks were evolved against a weak algorithm, AdaBoost, and the more powerful ones, Decision Tree, Random Forest and One-Class SVM. The evolutionary multiobjective optimisation approach developed earlier was adopted and improved for this problem using mul-

tiple objectives: maximise damage; minimise detection rate, minimise effort. We were able to find a range of attacks against all attacks; however, evading SVM while causing significant damage proved to be hard, demonstrating the value of this approach. On the other hand, when tested against AdaBoost, Decision Tree and Random Forest, we were capable of generating a range of attacks that could evade detection while causing significant damage.

The attacks generated show that if an adversary without detailed knowledge of the plant manages to compromise the process signals, they are not very likely to cause damage and evade detection as they will not easily be able to establish what signal(s) to attack, how to perform the attack, and when to carry out the attack. However, the results also show that a targeted attacker with access to some process level information will be able to generate attacks that do have an effect and that are hard to detect.

9.1.1 Main Contributions

This thesis has proposed and evaluated an effective and efficient methodology that can be used to identify vulnerabilities in industrial processes and intrusion detection mechanisms. Using this, security engineers can design better security mechanisms, both ab inito, and during system operation . The main contributions of our work are:

- We extended the implementation of a complex nonlinear chemical benchmark, the TE model, with attacks that facilitate the study of plant security. We studied the performance of the TE process on single attacks in terms of potential consequences.

- We have developed an approach based on evolutionary algorithms, evolutionary strategies, and multiobjective optimisation that permits security engineers proactively to identify the vulnerabilities of the process and we compared it to random search and single objective genetic algorithms. Because it is a particularly promising approach a wide range of experiments were carried out to evaluate the performance of the developed evolutionary multiobjective opti-

misation utilising the selection methods of two of widely used multiobjective evolutionary algorithms. As a result, we identified the most vulnerable combinations of sensors and actuators in the TE model.

- We investigated existing detection models, and identified weakness in these studies. However we subsequently developed a range of novel IDS methods that could improve the detection of attacks. As part of this, we developed datasets that could be used to train, validate and test detection.

- Finally, we adapted and applied our approach to develop attacks against IDS methods to identify remaining vulnerabilities in the plant and in the detection method.

9.2 Future work

In this section, we discuss some future directions for research.

The existing implementation of the TE model in MATLAB is too slow to be ideal for stochastic population-based optimisation algorithms like evolutionary algorithms and, therefore, our experiments were limited by this. This was particularly problematic when generating attacks against a system with IDS as the algorithms required more time converge properly. The results obtained show our evolutionary multiobjective optimisation is able to search the space efficiently, and give estimates of potential damage; however, future work is necessary to test how generalisable the results are. Migrating the code to another language, for example C++ or Python, may improve the performance, as well as providing greater flexibility in terms of implementing new attacks and integrating the model into a Hardware-in-the-loop (HIL) simulation with network communication protocols.

One of the most common problems of evolutionary algorithms is premature convergence, and we experienced this problem in some of our experiments. In respect of this problem, our approach might be further improved by analysing some of the more advanced techniques in the evolutionary computation literature aimed at improving the diversity of the population. Future work is also needed to investigate the benefit of seeding the initial population. We observed we were able to

find attacks against Random Forest when we seed the initial population using some of the best individuals obtained from a weaker classifier, the single Decision Tree. However, validation of the benefits of seeding requires a more comprehensive study, which we plan to address in future work. We will also investigate how other evolutionary multiobjective optimisation techniques based on aggregation-based and indicator-based algorithms will perform against the Pareto-based algorithms developed in this thesis.

The unavailability of benchmark models is a major obstacle for ICS security research. Testing our approach on other benchmark problems would help to verify and validate our approach but, unfortunately, there are few benchmark models that are available. One possible alternative would be the Vinyl Acetate Monomer plant [309]. We will investigate this further, but this is similar in structure to TE, and it is also a chemical batch process. We are currently working on building a physical microgrid testbed, and we hope to verify and validate the performance of our approach on a different type of process with physical and network components.

Unsupervised anomaly-based intrusion detection systems based on deep learning can be more effective than conventional methods. The success of these approaches depends on the design of the architecture and the choice of hyperparameters for the optimisation algorithm. A possible future approach is to use evolutionary algorithms to automatically tune the best hyperparameters for a given dataset. A more advanced method, based on genetic algorithms, was also recommended by deep learning researchers; this relies on population-based training of neural networks [310].

The TE model has six modes of operation with different G/H mass ratios designed to produce different production outputs to meet market demands. The experiments in this thesis were carried using a mode 1; future work should study the impact of changing modes on the detection methods. An additional area of future research is to design common metrics that can be used by industry to quantify rapidity of detection.

Future work will investigate if it is possible for an adversary with little prior

knowledge to design targetted attacks against the process using some of the new learning techniques in deep learning, in particular the application of the generative adversarial networks in learning the behaviour of the IDS.

Bibliography

[1] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," Technical Report, CISCO Internet Business Solutions Group (IBSG), 2011.

[2] D. Peterson, "Unsolicited Response Podcast Number 2 Bob Radvanovsky on Project Shine," Technical Report, Digital Bond Inc, 2012.

[3] L. O. M. Nicolas Falliere and E. Chien, "W32.Stuxnet Dossier (Version 1.4)," White Paper, Symantec Security Response, 2011.

[4] K. Zetter, "Meet 'Flame', The Massive Spy Malware Infiltrating Iranian Computers." Available: `http://www.wired.com/2012/05/flame/`, May 2012. Last Accessed: 15/04/2019.

[5] Kaspersky Lab, "Gauss: Abnormal Distribution," White Paper, 2012.

[6] E. Chien and G. O'Gorman, "The Nitro Attacks," White Paper, Symantec Security Response, 2011.

[7] Symantec, "W32.Duqu: The Precursor to the Next Stuxnet (Version 1.4)," White Paper, Symantec Security Response, 2011.

[8] R. Dorf C. and R. Bishop H., *Modern Control Systems, Twelfth Edition*. Pearson, 2011.

[9] K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams, and A. Hahn, "Guide to Industrial Control Systems (ICS) Security," Technical Report, National Institute of Standards and Technology, AGaithersburg, MD, USA, 2014.

[10] T.-C. Yang, "Networked control system: a brief survey," *Control Theory and Applications, IEE Proceedings*, vol. 153, pp. 403–412, July 2006.

[11] G. Schickhuber and O. McCarthy, "Distributed fieldbus and control network systems," *Computing Control Engineering Journal*, vol. 8, pp. 21–32, Feb 1997.

[12] N. Tuptuk and S. Hailes, "Security of smart manufacturing systems," *Journal of Manufacturing Systems*, vol. 47, pp. 93 – 106, 2018.

[13] N. Tuptuk and S. Hailes, "The cyberattack on Ukraine's power grid is a warning of what's to come." Available: `https://theconversation.com/the-cyberattack-on-ukraines-power-grid-is-a-warning-of-whats-to-come-52832`, Jan 2016. Last Accessed: 15/04/2019.

[14] Bundesamt für Sicherheit in der Informationstechnik (BSI), "Die Lage der IT-Sicherheit in Deutschland 2014," Technical Report, 2014.

[15] F-Secure, "Havex Hunts For ICS/SCADA Systems." Available: `https://www.f-secure.com/weblog/archives/00002718.html`, Jun 2014. Last Accessed: 04/04/2017.

[16] Gartner Inc, "Gartner Press Release: Gartner Says 4.9 Billion Connected Things Will Be in Use in 2015," Press Releases, 2014.

[17] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[18] M. Krotofil and J. Larsen, "Rocking the pocket book: Hacking chemical plants for competition and extortion," White Paper, DefCon 23, 2015.

[19] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?," in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, (New York, NY, USA), pp. 16–25, ACM, 2006.

[20] J. Downs and E. Vogel, "A plant-wide industrial process control problem," *Computers Chemical Engineering*, vol. 17, no. 3, pp. 245 – 255, 1993.

[21] Symantec, "Symantec Intelligence Report: May 2015," Intelligence Report, 2015.

[22] Oxford University Press, "Attack." Available: `https://en.oxforddictionaries.com/definition/attack`. Last Accessed: 10/04/2019.

[23] National Institute of Standards and Technology, "Minimum security requirements for federal information and information systems." Available: `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.200.pdf`, March 2006. Last Accessed: 10/04/2019.

[24] "Security and Privacy Controls for Federal Information Systems and institutions." Available: `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf`, April 2013. Last Accessed: 10/04/2019.

[25] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *Security Privacy, IEEE*, vol. 9, pp. 49–51, May 2011.

[26] D. Beresford, "Exploiting Siemens Simatic S7 PLCs," White Paper, NSS Labs, 2011.

[27] B. Kerbs, "Cyber incident blamed for nuclear power plant shutdown." Available: `http://www.washingtonpost.com/wp-dyn/content/article/2008/06/05/AR2008060501958.html`, Jun 2008. Last Accessed: 23/03/2015.

[28] The Register, "Data storm blamed for nuclear plant shutdown." Available: `http://www.theregister.co.uk/2007/05/21/alabama_nuclear_plant_shutdown/`, May 2007. Last Accessed: 10/02/2018.

[29] M. Giles, "Triton is the world's most murderous malware, and it's spreading." Available: https://www.technologyreview.com/s/613054/cybersecurity-critical-infrastructure-triton-malware, March 2019. Last Accessed: 14/05/2019.

[30] Dragos Inc, "TRISIS Malware Analysis of Safety System Targeted Malware, Version 1.20171213." Available: https://dragos.com/wp-content/uploads/TRISIS-01.pdf, December 2017. Last Accessed: 04/06/2018.

[31] R. Thomas, "Triton malware spearheads latest generation of attacks on industrial systems." Available: https://securingtomorrow.mcafee.com/other-blogs/mcafee-labs/triton-malware-spearheads, December 2018. Last Accessed: 14/05/2019.

[32] P. Paganini, "Researcher found Wind turbines and solar systems vulnerable worldwide." Available: https://securityaffairs.co/wordpress/37792/hacking/wind-turbines-hacking.html, June 2015. Last Accessed: 15/04/2018.

[33] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-15-076-01): XZERES 442SR Wind Turbine Vulnerability." Available: https://ics-cert.us-cert.gov/advisories/ICSA-15-076-01, Mar 2015. Last Accessed: 10/01/2019.

[34] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-15-160-02): Sinapsi eSolar Light Plaintext Passwords Vulnerability." Available: https://ics-cert.us-cert.gov/advisories/ICSA-15-160-02, Jun 2015. Last Accessed: 10/01/2019.

[35] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-15-132-02) Rockwell Automation RSView32 Weak Encryption Algorithm on Passwords." Available: https://ics-cert.us-cert.

gov/advisories/ICSA-15-132-02, May 2015. Last Accessed: 10/01/2019.

[36] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-15-167-01) GarrettCom Magnum Series Devices Vulnerabilities." Available: https://ics-cert.us-cert.gov/advisories/ICSA-15-167-01, Jun 2015. Last Accessed: 10/01/2019.

[37] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-15-013-01) Siemens SIMATIC WinCC Sm@rtClient iOS Application Authentication Vulnerabilities." Available: https://ics-cert.us-cert.gov/advisories/ICSA-15-013-01, Jan 2015. Last Accessed: 10/01/2019.

[38] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-15-013-03) Phoenix Contact Software ProConOs and MultiProg Authentication Vulnerability." Available: https://ics-cert.us-cert.gov/advisories/ICSA-15-013-03, Jan 2015. Last Accessed: 10/01/2019.

[39] Symantec, "Dragonfly: Cyberespinoage Attacks Against Energy Suppliers," Symantec Security Response, Jul 2014.

[40] L. of Cryptography and S. S. (CrySyS), "Duqu: A Stuxnet-like malware found in the wild v0.93," Technical Report, 2011.

[41] P. F. Roberts, "Zotob, PnP Worms Slam 13 DaimlerChrysler Plants." Available: http://www.eweek.com/c/a/Security/Zotob-PnP-Worms-Slam-13-DaimlerChrysler-Plants, Aug 2005. Last Accessed: 13/01/2019.

[42] Symantec, "W32.Zotob.E." Available http://www.symantec.com/security_response/writeup.jsp?docid=2005-081615-4443-99, Feb 2007. Last Accessed: 13/04/2019.

[43] BBC, "Zotob virus writers face prison." Available: http://news.bbc.co.uk/1/hi/technology/5345404.stm, Sep 2006. Last Accessed: 13/04/2019.

[44] BBC, "Two detained for US computer worm." Available: http://news.bbc.co.uk/1/hi/technology/4189996.stm, Aug 2005. Last Accessed: 14/04/2019.

[45] The Wire, "Hackers Shut Down a Tunnel Road in Israel." Available: http://www.thewire.com/global/2013/10/hackers-shut-down-tunnel-road-israel/70983, Oct 2013. Last Accessed: 14/04/2019.

[46] Cryptome Org, "Hackers Shut Down a Tunnel Road in Israel." Available: http://cryptome.org/2013/05/sea-haifa-hack.htm, May 2013. Last Accessed: 14/04/2019.

[47] The Register, "Disgruntled techie attempts californian power blackout." Available: http://www.theregister.co.uk/2007/04/20/terrorists_among_us_flee_flee, Nov 2007. Last Accessed: 14/04/2019.

[48] ComputerWeekly, "Cyber attack on US shipping exploited known security hole: teenager accused." Available: https://www.computerweekly.com/news/2240052986/Cyber-attack-on-US-shipping-exploited-known-security-hole-teenager-accused, Oct 2003. Last Accessed: 16/04/2019.

[49] SANS ICS, Industrial Control Systems and Electricity Information Sharing and Analysis Center, "TLP: WhiteAnalysis of the Cyber Attack on the Ukrainian Power Grid Defense Use Case," Technical Report, 2016.

[50] R. Khan, P. Maynard, K. McLaughlin, D. Laverty, and S. Sezer, "Threat analysis of blackenergy malware for synchrophasor based real-time control and

monitoring in smart grid," in *Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research 2016*, ICS-CSR '16, (Swindon, UK), pp. 1–11, BCS Learning & Development Ltd., 2016.

[51] Dragos Inc, "CRASHOVERRIDE Analyzing the Threat to Electric Grid Operations Version 2.20170613." Available: `https://dragos.com/wp-content/uploads/CrashOverride-01.pdf`, June 2017. Last Accessed: 04/06/2018.

[52] The Telegraph, "CIA plot led to huge blast in Siberian gas pipeline." Available: `https://www.telegraph.co.uk/news/worldnews/northamerica/usa/1455559/CIA-plot-led-to-huge-blast-in-Siberian-gas-pipeline.html`, Feb 2004. Last Accessed: 04/06/2018.

[53] Symantec, "Flamer: Highly Sophisticated and Discreet Threat Targets the Middle East." Available: `http://www.symantec.com/connect/blogs/flamer-highly-sophisticated-and-discreet-threat-targets-middle-east`, May 2012. Last Accessed: 13/05/2018.

[54] Symantec, "The Shamoon Attacks." Available: `https://www.symantec.com/connect/blogs/shamoon-attacks`, Aug 2012. Last Accessed: 13/05/2018.

[55] B. Krebs, "Chinese Hackers Blamed for Intrusion at Energy Industry Giant Telvent." Available: `https://krebsonsecurity.com/2012/09/chinese-hackers-blamed-for-intrusion-at-energy-industry-giant-telvent/`, Sep 2012. Last Accessed: 10/03/2019.

[56] Schneider Electric USA Inc., "Oasys product family overview," 2012.

[57] The Registry, "Russia welcomes hack attacks." Available: `http://www.theregister.co.uk/2000/04/27/russia_welcomes_hack_attacks/`, Apr 2000. Last Accessed: 10/04/2019.

[58] ControlGlobal, "Illinois Water Hack Causes Confusion." Available: http://www.controlglobal.com/industrynews/2011/illinois-water-hack-causes-confusion/, Dec 2011. Last Accessed: 08/03/2018.

[59] ThreatPost, "Hacker says texas town used three character password to secure internet facing scada system." Available: https://threatpost.com/hacker-says-texas-town-used-three-character-password-secure-internet-facing-scada-system-11201/75914, Nov 2011.

[60] M. Abrams and J. Weiss, "Malicious Control System Cyber Security Attack Case Study–Maroochy Water Services, Australia." Available: https://www.mitre.org/sites/default/files/pdf/08_1145.pdf, Jul 2008. Last Accessed: 13/04/2019.

[61] The Registry, "Electrical supe charged with damaging California canal system." Available: http://www.theregister.co.uk/2007/11/30/canal_system_hack/, Nov 2007. Last Accessed: 14/04/2019.

[62] BBC, "Japan defence firm Mitsubishi Heavy in cyber attack." Available: http://www.bbc.co.uk/news/world-asia-pacific-14982906, Sep 2011. Last Accessed: 08/03/2018.

[63] S. Ishimaru, "New activity of the Blue Termite APT." Available: https://securelist.com/blog/research/71876/new-activity-of-the-blue-termite-apt/, Aug 2015.

[64] BBC, "Fines fraud hits Italian drivers." Available: http://news.bbc.co.uk/1/hi/world/europe/7862893.stm, Jan 2019. Last Accessed: 14/04/2019.

[65] The Telegraph, "'Zombies ahead', warns electronic road sign." Available:http://www.telegraph.co.uk/news/newstopics/

howaboutthat/4518092/Zombies-ahead-warns-electronic-road-sign.html, Feb 2009. Last Accessed: 18/04/2019.

[66] Los Angeles Times, "Key signals targeted, officials say." Available: http://articles.latimes.com/2007/jan/09/local/me-trafficlights9, Jan 2007. Last Accessed: 14/04/2019.

[67] GeekCom, "LA traffic engineers plead guilty to hacking traffic signals." Available: http://www.geek.com/news/la-traffic-engineers-plead-guilty-to-hacking-traffic-signals-621672/, Nov 2008. Last Accessed: 14/04/2019.

[68] The Computer World, "Hackers break into water system network." Available: http://www.computerworld.com/article/2547938/security0/hackers-break-into-water-system-network.html, Oct 2006. Last Accessed: 10/02/2018.

[69] T. Telegraph, "Schoolboy hacks into city's tram system." Available: http://www.telegraph.co.uk/news/worldnews/1575293/Schoolboy-hacks-into-citys-tram-system.html, Jan 2008. Last Accessed: 14/04/2019.

[70] Information Week, "Computer virus brings down train signals." Available: http://www.informationweek.com/computer-virus-brings-down-train-signals/d/d-id/1020446, Aug 2003. Last Accessed: 11/02/2018.

[71] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm," *Security Privacy, IEEE*, vol. 1, pp. 33–39, July 2003.

[72] SecurityFocus, "Slammer worm crashed ohio nuke plant network." Available: http://www.securityfocus.com/news/6767, Aug 2003. Last Accessed: 15/02/2019.

[73] BBC, "Police warning after drug traffickers' cyber-attack." Available: http://www.bbc.co.uk/news/world-europe-24539417, Oct 2013. Last Accessed: 14/04/2019.

[74] INTERPOL, "Against Organized Crime: INTERPOL Trafficking and Counterfeiting Casebook 2014," 2014.

[75] SecurityWeek, "Simulation Shows Threat of Ransomware Attacks on ICS." Available: https://www.securityweek.com/simulation-shows-threat-ransomware-attacks-ics, Feb 2017. Last Accessed: 15/04/2018.

[76] BBC, "Scrap metal regulations urged to combat theft." Available: http://www.bbc.co.uk/news/uk-15704549, Nov. 2011. Last Accessed: 15/04/2018.

[77] BBC, "Substation copper thieves 'putting lives at risk'." Available: http://news.bbc.co.uk/1/hi/england/8696724.stm, Mar. 2010. Last Accessed: 15/04/2018.

[78] S. Lipscombe and O. Bennett, "House of Commons Briefing papers: Metal theft (SN/HA/6150)," Research Briefings, Jul 2012.

[79] M. Joyce, *Digital Activism Decoded: The New Mechanics of Change*. Idebate Press, 2010.

[80] The Times, "'Anonymous' hackers join Iran protests." Available: https://www.thetimes.co.uk/article/anonymous-hackers-join-iran-protests-6qwvhw2j5h0, Feb 2011. Last Accessed: 12/04/2018.

[81] The Sydney Morning Herald, "Operation Titstorm: hackers bring down government websites." Available:https://www.smh.com.au/technology/operation-titstorm-hackers-bring-

down-government-websites-20100210-nqku.html, Feb 2010. Last Accessed: 12/04/2018.

[82] Redmond, "UPDATED: Anonymous/LulzSec Hack Timeline." Available: https://redmondmag.com/articles/2011/06/27/timeline-of-anonymous-lulzsec-hacks.aspx, Aug 2011. Last Accessed: 12/04/2018.

[83] BBC, "Hackers hit Italian cyber-police." Available: HackershitItaliancyber-police, July 2011. Last Accessed: 12/04/2018.

[84] C. P. Pfleeger, *Reflections on the Insider Threat*, pp. 5–16. Boston, MA: Springer US, 2008.

[85] M. Collins, M. Theis, R. Trzeciak, J. Strozer, J. Clark, D. Costa, T. Cassidy, M. Albrethsen, and A. Moore, "Common Sense Guide to Mitigating Insider Threats, Fifth Edition," 12 2016.

[86] U.S. Secret Service, Software Engineering Institute CERT Program at Carnegie Mellon University and Price Waterhouse Cooper, "2014 US State of Cybercrime Survey, CSO Magazine," Apr 2014.

[87] M. Keeney, E. Kowalski, D. Cappelli, A. Moore, T. Shimeall, and S. Rogers, "Insider threat study: Computer system sabotage in critical infrastructure sectors," tech. rep., Cornegie Mellon, Software Engineering Institute, 2005.

[88] HotforSecurity, "One in Three Employees Would Sell All Company Secrets for the Right Price." Available:http://www.hotforsecurity.com/blog/one-in-three-employees-would-sell-all-company-secrets-for-the-right-price-12392.html, Aug 2015. Last Accessed: 10/04/2019.

[89] R. Spreitzer, V. Moonsamy, T. Korak, and S. Mangard, "Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices," *IEEE Communications Surveys Tutorials*, vol. 20, pp. 465–488, Firstquarter 2018.

[90] N. Tuptuk and S. Hailes, "Covert channel attacks in pervasive computing," in *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 236–242, 2015.

[91] L. Bilge and T. Dumitraundefined, "Before We Knew It: An Empirical Study of Zero-Day Attacks in the Real World," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, (New York, NY, USA), p. 833–844, Association for Computing Machinery, 2012.

[92] "Scikit-learn Machine Learning in Python: User Guide (version 0.24.0)." Available: `https://scikit-learn.org/stable/user_guide.html`, 2019. Last Accessed: 14/05/2019.

[93] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Comput. Surv.*, vol. 46, pp. 55:1–55:29, Mar. 2014.

[94] P. Oman and M. Phillips, "Intrusion Detection and Event Monitoring in SCADA Networks," in *Critical Infrastructure Protection* (E. Goetz and S. Shenoi, eds.), vol. 253 of *IFIP International Federation for Information Processing*, pp. 161–173, Springer US, 2008.

[95] U. Premaratne, J. Samarabandu, T. Sidhu, R. Beresh, and J.-C. Tan, "An Intrusion Detection System for IEC61850 Automated Substations," *Power Delivery, IEEE Transactions on*, vol. 25, pp. 2376–2383, Oct 2010.

[96] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for scada networks," in *Proceedings of the SCADA Security Scientific Symposium*, (Miami Beach, Florida), Jan. 2007.

[97] A. Valdes and S. Cheung, "Intrusion Monitoring in Process Control Systems," in *2009 42nd Hawaii International Conference on System Sciences*, pp. 1–7, Jan 2009.

[98] N. Goldenberg and A. Wool, "Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 63 – 75, 2013.

[99] M. Caselli, E. Zambon, and F. Kargl, "Sequence-aware intrusion detection in industrial control systems," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, CPSS '15, (New York, NY, USA), pp. 13–24, ACM, 2015.

[100] O. Linda, T. Vollmer, and M. Manic, "Neural network based intrusion detection system for critical infrastructures," in *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pp. 1827–1834, June 2009.

[101] P. Dussel, C. Gehl, P. Laskov, J.-U. Buber, C. Stormann, and J. Kastner, "Cyber-critical infrastructure protection using real-time payload-based anomaly detection," in *Critical Information Infrastructures Security* (E. Rome and R. Bloomfield, eds.), vol. 6027 of *Lecture Notes in Computer Science*, pp. 85–97, Springer Berlin Heidelberg, 2010.

[102] H. Hadeli, R. Schierholz, M. Braendle, and C. Tuduce, "Leveraging determinism in industrial control systems for advanced anomaly detection and reliable security configuration," in *Emerging Technologies Factory Automation, 2009. ETFA 2009. IEEE Conference on*, pp. 1–8, Sept 2009.

[103] A. Paul, F. Schuster, and H. König, "Towards the Protection of Industrial Control Systems: Conclusions of a Vulnerability Analysis of Profinet IO," in *Proceedings of the 10th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA'13, (Berlin, Heidelberg), pp. 160–176, Springer-Verlag, 2013.

[104] D. Hadziosmanovic, L. Simionato, D. Bolzoni, E. Zambon, and S. Etalle, *N-Gram against the Machine: On the Feasibility of the N-Gram Network Analysis for Binary Protocols*, pp. 354–373. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[105] S. Shin, T. Kwon, G. Y. Jo, Y. Park, and H. Rhy, "An Experimental Study of Hierarchical Intrusion Detection for Wireless Industrial Sensor Networks," *IEEE Transactions on Industrial Informatics*, vol. 6, pp. 744–757, Nov 2010.

[106] Y. Zhang, L. Wang, W. Sun, R. Green, and M. Alam, "Artificial immune system based intrusion detection in a distributed hierarchical network architecture of smart grid," in *Power and Energy Society General Meeting, 2011 IEEE*, pp. 1–8, July 2011.

[107] Y. Zhang, L. Wang, W. Sun, R. Green, and M. Alam, "Distributed Intrusion Detection System in a Multi-Layer Network Architecture of Smart Grids," *Smart Grid, IEEE Transactions on*, vol. 2, pp. 796–808, Dec 2011.

[108] S. Pan, T. Morris, and U. Adhikari, "Developing a hybrid intrusion detection system using data mining for power systems," *IEEE Transactions on Smart Grid*, vol. 6, pp. 3104–3113, Nov 2015.

[109] W. Gao, T. Morris, B. Reaves, and D. Richey, "On scada control system command and response injection and intrusion detection," in *eCrime Researchers Summit (eCrime), 2010*, pp. 1–9, Oct 2010.

[110] W. Jardine, S. Frey, B. Green, and A. Rashid, "SENAMI: Selective Non-Invasive Active Monitoring for ICS Intrusion Detection," in *Proceedings of the 2Nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, CPS-SPC '16, (New York, NY, USA), pp. 23–34, ACM, 2016.

[111] H. R. Ghaeini and N. O. Tippenhauer, "HAMIDS: Hierarchical Monitoring Intrusion Detection System for Industrial Control Systems," in *Proceedings of the 2Nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, CPS-SPC '16, (New York, NY, USA), pp. 103–111, ACM, 2016.

[112] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *Critical Information Infrastructures Security* (G. Havarneanu, R. Setola, H. Nassopoulos, and S. Wolthusen, eds.), (Cham), pp. 88–99, Springer International Publishing, 2017.

[113] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun, "Anomaly detection for a water treatment system using unsupervised machine learning," *CoRR*, vol. abs/1709.05342, 2017.

[114] M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy, CPS-SPC@CCS 2018, Toronto, ON, Canada, October 19, 2018*, pp. 72–83, 2018.

[115] G. Sabaliauskaite, G. Ng, J. Ruths, and A. Mathur, "A comprehensive approach, and a case study, for conducting attack detection experiments in cyber–physical systems," *Robotics and Autonomous Systems*, vol. 98, pp. 174 – 191, 2017.

[116] T. R. McEvoy and S. D. Wolthusen, "Trouble brewing: Using observations of invariant behavior to detect malicious agency in distributed control systems," in *Proceedings of the 4th International Conference on Critical Information Infrastructures Security*, CRITIS'09, (Berlin, Heidelberg), pp. 62–72, Springer-Verlag, 2010.

[117] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel, "Through the eye of the plc: Semantic security monitoring for industrial processes," in *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC '14, (New York, NY, USA), pp. 126–135, ACM, 2014.

[118] P. Filonov, A. Lavrentyev, and A. Vorontsov, "Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model," *CoRR*, vol. abs/1612.06676, 2016.

[119] A. A. Cardenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks Against Process Control Systems: Risk Assessment, Detection, and Response," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, (New York, NY, USA), pp. 355–366, ACM, 2011.

[120] A. Keliris, H. Salehghaffari, B. Cairl, P. Krishnamurthy, M. Maniatakos, and F. Khorrami, "Machine learning-based defense against process-aware attacks on industrial control systems," in *2016 IEEE International Test Conference (ITC)*, pp. 1–10, Nov 2016.

[121] J. Chen and C.-M. Liao, "Dynamic process fault monitoring based on neural network and pca," *Journal of Process Control*, vol. 12, no. 2, pp. 277 – 289, 2002.

[122] I. Kiss, B. Genge, and P. Haller, "A clustering-based approach to detect cyber attacks in process control systems," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pp. 142–148, July 2015.

[123] P. Filonov, F. Kitashov, and A. Lavrentyev, "RNN-based Early Cyber-Attack Detection for the Tennessee Eastman Process," *CoRR*, vol. abs/1709.02232, 2017.

[124] N. L. Ricker, "Tennessee Eastman Challenge Archive." Available: `http://depts.washington.edu/control/LARRY/TE/download.html#Multiloop`, Dec 1998. Last Updated: Jan 2015.

[125] Braatzgroup, The MIT Process Systems Engineering Laboratory, "Tennessee Eastman Problem Simulation Data." Available: `http://web.mit.edu/braatzgroup/links.html`. Last Accessed: 15/09/2018.

[126] A. Raich and A. Çinar, "Multivariate statistical methods for monitoring continuous processes: assessment of discrimination power of disturbance models

and diagnosis of multiple disturbances," *Chemometrics and Intelligent Laboratory Systems*, vol. 30, no. 1, pp. 37 – 48, 1995. InCINC '94 Selected papers from the First International Chemometrics Internet Conference.

[127] A. Raich and A. Çinar, "Diagnosis of process disturbances by statistical distance and angle measures," *Computers Chemical Engineering*, vol. 21, no. 6, pp. 661 – 673, 1997.

[128] M. Kano, K. Nagao, S. Hasebe, I. Hashimoto, H. Ohno, R. Strauss, and B. Bakshi, "Comparison of statistical process monitoring methods: application to the eastman challenge problem," *Computers Chemical Engineering*, vol. 24, no. 2, pp. 175 – 181, 2000.

[129] E. L. Russell, L. H. Chiang, and R. D. Braatz, "Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 51, no. 1, pp. 81 – 93, 2000.

[130] W. Lin, Y. Qian, and X. Li, "Nonlinear dynamic principal component analysis for on-line process monitoring and diagnosis," *Computers Chemical Engineering*, vol. 24, no. 2, pp. 423 – 429, 2000.

[131] G. Chen and T. J. McAvoy, "Predictive on-line monitoring of continuous processes," *Journal of Process Control*, vol. 8, no. 5, pp. 409 – 420, 1998. ADCHEM '97 IFAC Symposium: Advanced Control of Chemical Processes.

[132] F. Akbaryan and P. Bishnoi, "Fault diagnosis of multivariate systems using pattern recognition and multisensor data analysis technique," *Computers Chemical Engineering*, vol. 25, no. 9, pp. 1313 – 1339, 2001.

[133] A. Bakdi and A. Kouadri, "A new adaptive pca based thresholding scheme for fault detection in complex systems," *Chemometrics and Intelligent Laboratory Systems*, vol. 162, pp. 83 – 93, 2017.

[134] L. H. Chiang, E. L. Russell, and R. D. Braatz, "Fault diagnosis in chemical processes using fisher discriminant analysis, discriminant partial least squares, and principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 2, pp. 243 – 252, 2000.

[135] M. Misra, H. Yue, S. Qin, and C. Ling, "Multivariate process monitoring and fault diagnosis by multi-scale pca," *Computers Chemical Engineering*, vol. 26, no. 9, pp. 1281 – 1293, 2002.

[136] L. H. Chiang, M. E. Kotanchek, and A. K. Kordon, "Fault diagnosis based on fisher discriminant analysis and support vector machines," *Computers Chemical Engineering*, vol. 28, no. 8, pp. 1389 – 1401, 2004.

[137] Y. Zhang, "Enhanced statistical analysis of nonlinear processes using KPCA, KICA and SVM," *Chemical Engineering Science*, vol. 64, no. 5, pp. 801 – 811, 2009.

[138] A. Beghi, L. Cecchinato, C. Corazzol, M. Rampazzo, F. Simmini, and G. Susto, "A one-class svm based tool for machine learning novelty detection in hvac chiller systems," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 1953 – 1958, 2014. 19th IFAC World Congress.

[139] S. Mahadevan and S. L. Shah, "Fault detection and diagnosis in process data using one-class support vector machines," *Journal of Process Control*, vol. 19, no. 10, pp. 1627 – 1639, 2009.

[140] X. Gao and J. Hou, "An improved svm integrated gs-pca fault diagnosis approach of tennessee eastman process," *Neurocomputing*, vol. 174, pp. 906 – 911, 2016.

[141] H. Chen, P. Tiňo, and X. Yao, "Cognitive fault diagnosis in Tennessee Eastman Process using learning in the model space," *Computers Chemical Engineering*, vol. 67, pp. 33 – 42, 2014.

[142] H. Chen, P. Tiňo, A. Rodan, and X. Yao, "Learning in the model space for cognitive fault diagnosis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, pp. 124–136, Jan 2014.

[143] L. Martí, N. S. Pi, J. M. M. López, and A. C. B. Garcia, "Anomaly detection based on sensor data in petroleum industry applications," in *Sensors*, 2015.

[144] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *CoRR*, vol. abs/1607.00148, 2016.

[145] F. Lv, C. Wen, Z. Bao, and M. Liu, "Fault diagnosis based on deep learning," in *2016 American Control Conference (ACC)*, pp. 6851–6856, July 2016.

[146] H. Zhao, S. Sun, and B. Jin, "Sequential fault diagnosis based on lstm neural network," *IEEE Access*, vol. 6, pp. 12929–12939, 2018.

[147] iTrust (Centre for Research in Cyber Security), Signapore University of Technology and Design, "Secure Water Treatment (SWaT) Dataset." Available: `https://itrust.sutd.edu.sg/research/dataset/dataset_characteristics/#swat`, 2018.

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

[149] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[150] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[151] M. A. Arostegui, J. N. Kadipasaoglu, and B. M.Khumawala, "An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems," *International Journal of Production Economics*, vol. 103, no. 2, pp. 742 – 754, 2006.

[152] K. Mrugala, N. Tuptuk, and S. Hailes, "Evolving attackers against wireless sensor networks," in *GECCO 2016 Companion Volume*, (Denver, USA), p. pp306, ACM, 20-24 July 2016.

[153] K. Mrugala, N. Tuptuk, and S. Hailes, "Evolving attackers against wireless sensor networks using genetic programming," *IET Wireless Sensor Systems*, vol. 7, no. 4, pp. 113–122, 2017.

[154] D. B. Fogel, "What is evolutionary computation?," *IEEE Spectrum*, vol. 37, pp. 26–32, Feb 2000.

[155] O. Flasch, M. Friese, K. Vladislavleva, T. Bartz-Beielstein, O. Mersmann, B. Naujoks, J. Stork, and M. Zaefferer, *Comparing Ensemble-Based Forecasting Methods for Smart-Metering Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[156] K. Seo, B. Hyeon, S. Hyun, and Y. Lee, *Genetic Programming-Based Model Output Statistics for Short-Range Temperature Prediction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[157] E. Riva Sanseverino, M. L. Di Silvestre, and R. Gallea, *Pareto-optimal Glowworm Swarms Optimization for Smart Grids Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[158] M. J. Eppstein and C. B. Ogbunugafor, "Quantifying deception: A case study in the evolution of antimicrobial resistance," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*, pp. 101–108, 2016.

[159] S. Perez-Carabaza, E. Besada-Portas, J. A. Lopez-Orozco, and J. M. de la Cruz, "A Real World Multi-UAV Evolutionary Planner for Minimum Time Target Detection," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, (New York, NY, USA), pp. 981–988, ACM, 2016.

[160] M. Braun, T. Dengiz, I. Mauser, and H. Schmeck, *Comparison of Multi-objective Evolutionary Optimization in Smart Building Scenarios*. Cham: Springer International Publishing, 2016.

[161] J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, eds., *Proceedings of the 1st Annual Conference on Genetic Programming*, (Cambridge, MA, USA), MIT Press, 1996.

[162] D. Wagner and P. Soto, "Mimicry Attacks on Host-based Intrusion Detection Systems," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, (New York, NY, USA), pp. 255–264, ACM, 2002.

[163] K. M. C. Tan, K. S. Killourhy, and R. A. Maxion, "Undermining an Anomaly-based Intrusion Detection System Using Common Exploits," in *Proceedings of the 5th International Conference on Recent Advances in Intrusion Detection*, RAID'02, (Berlin, Heidelberg), pp. 54–73, Springer-Verlag, 2002.

[164] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-Nonself Discrimination in a Computer," in *Proceedings of the 1994 IEEE Symposium on Security and Privacy*, SP '94, (Washington, DC, USA), pp. 202–, IEEE Computer Society, 1994.

[165] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, SP '96, (Washington, DC, USA), pp. 120–, IEEE Computer Society, 1996.

[166] MIT Lincoln Laboratory, "DARPA Intrusion Detection Data Sets." Available: `http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/`, 2000.

[167] W. Li, "Using Genetic Algorithm for network intrusion detection," in *In Proceedings of the United States Department of Energy Cyber Security Group 2004 Training Conference*, pp. 24–27, 2004.

[168] T. Xia, G. Qu, S. Hariri, and M. Yousif, "An efficient network intrusion detection method based on information theory and genetic algorithm," in *PCCC 2005. 24th IEEE International Performance, Computing, and Communications Conference, 2005.*, pp. 11–17, April 2005.

[169] Chittur, Adhitya, and A. Chittur, "Model generation for an intrusion detection system using genetic algorithms," Dissertation, 2001.

[170] T. Vollmer, J. Alves-Foss, and M. Manic, "Autonomous rule creation for intrusion detection," in *2011 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, pp. 1–8, April 2011.

[171] A. Ojugo, A. Eboka, O. Okonta, R. Yoro, and F. Aghware, "Genetic algorithm rule-based intrusion detection system (GAIDS)," *Journal of Emerging Trends in Computing and Information Sciences*, 2012.

[172] M. S. Hoque, M. A. Mukit, and M. A. N. Bikas, "An Implementation of Intrusion Detection System Using Genetic Algorithm," *CoRR*, vol. abs/1204.1336, 2012.

[173] P. A. Diaz-gomez, I. D. Sistemas, and D. F. Hougen, "Improved off-line intrusion detection using a genetic algorithm," in *in Proceedings of the Seventh International Conference on Enterprise Information Systems*, 2005.

[174] A. Goyal and C. Kumar, "GA-NIDS: A Genetic Algorithm based Network Intrusion Detection System," in *ElectricalEngineering Computer Science, North West University, Technical Report*, 2008.

[175] J. Budynek, E. Bonabeau, and B. Shargel, "Evolving computer intrusion scripts for vulnerability assessment and log analysis," in *Proceedings of the*

7th Annual Conference on Genetic and Evolutionary Computation, GECCO '05, (New York, NY, USA), pp. 1905–1912, ACM, 2005.

[176] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1993.

[177] W. Lu and I. Traore, "Detecting New Forms of Network Intrusion Using Genetic Programming," *Computational Intelligence*, vol. 20, no. 3, pp. 475–494, 2004.

[178] S. Pastrana, A. Orfila, and A. Ribagorda, "A functional framework to evade network ids," in *2011 44th Hawaii International Conference on System Sciences*, pp. 1–10, Jan 2011.

[179] C. Wallenta, J. Kim, P. J. Bentley, and S. Hailes, "Detecting interest cache poisoning in sensor networks using an artificial immune algorithm," *Applied Intelligence*, vol. 32, pp. 1–26, Feb 2010.

[180] H. G. Kayacik, M. Heywood, and N. Zincir-Heywood, "On evolving buffer overflow attacks using genetic programming," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, (New York, NY, USA), pp. 1667–1674, ACM, 2006.

[181] H. G. Kayacik, A. N. Zincir-Heywood, M. I. Heywood, and S. Burschka, "Generating mimicry attacks using genetic programming: A benchmarking study," in *2009 IEEE Symposium on Computational Intelligence in Cyber Security*, pp. 136–143, March 2009.

[182] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Evolutionary computation as an artificial attacker: generating evasion attacks for detector vulnerability testing," *Evolutionary Intelligence*, vol. 4, no. 4, pp. 243–266, 2011.

[183] S. Noreen, S. Murtaza, M. Z. Shafiq, and M. Farooq, "Evolvable malware," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary*

Computation, GECCO '09, (New York, NY, USA), pp. 1569–1576, ACM, 2009.

[184] G. Meng, Y. Xue, C. Mahinthan, A. Narayanan, Y. Liu, J. Zhang, and T. Chen, "Mystique: Evolving android malware for auditing anti-malware tools," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '16, (New York, NY, USA), pp. 365–376, ACM, 2016.

[185] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Parallel Problem Solving from Nature - PPSN VIII* (X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel, eds.), (Berlin, Heidelberg), pp. 832–842, Springer Berlin Heidelberg, 2004.

[186] A. Calleja, A. Martín, H. D. Menéndez, J. Tapiador, and D. Clark, "Picking on the family: Disrupting android malware triage by forcing misclassification," *Expert Systems with Applications*, vol. 95, pp. 113 – 126, 2018.

[187] E. Aydogan and S. Sen, "Automatic generation of mobile malwares using genetic programming," in *Applications of Evolutionary Computation* (A. M. Mora and G. Squillero, eds.), (Cham), pp. 745–756, Springer International Publishing, 2015.

[188] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on pdf malware classifiers," in *NDSS*, 2016.

[189] D. J. John, R. W. Smith, W. H. Turkett, D. A. Cañas, and E. W. Fulp, "Evolutionary based moving target cyber defense," in *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO Comp '14, (New York, NY, USA), pp. 1261–1268, ACM, 2014.

[190] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley, "Optimal security hardening using multi-objective optimization on attack tree models of networks," in

Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS '07, (New York, NY, USA), pp. 204–213, ACM, 2007.

[191] M. A. Potter and K. A. D. Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.

[192] J. D. Lohn, W. F. Kraus, and G. L. Haith, "Comparing a coevolutionary genetic algorithm for multiobjective optimization," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, vol. 2, pp. 1157–1162 vol.2, May 2002.

[193] L. Bull, "On coevolutionary genetic algorithms," *Soft Computing*, vol. 5, pp. 201–207, Jun 2001.

[194] G. Rush, D. R. Tauritz, and A. D. Kent, "Coevolutionary Agent-based Network Defense Lightweight Event System (CANDLES)," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, (New York, NY, USA), pp. 859–866, ACM, 2015.

[195] D. Garcia, A. E. Lugo, E. Hemberg, and U.-M. O'Reilly, "Investigating coevolutionary archive based genetic algorithms on cyber defense networks," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, (New York, NY, USA), pp. 1455–1462, ACM, 2017.

[196] E. D. de Jong, "The incremental pareto-coevolution archive," in *Genetic and Evolutionary Computation – GECCO 2004* (K. Deb, ed.), (Berlin, Heidelberg), pp. 525–536, Springer Berlin Heidelberg, 2004.

[197] T. Service, D. Tauritz, and W. Siever, "Infrastructure hardening: A competitive coevolutionary methodology inspired by neo-darwinian arms races," in *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, vol. 1, pp. 101–104, July 2007.

[198] T. Service and D. Tauritz, "Increasing infrastructure resilience through competitive coevolution," *New Mathematics and Natural Computation*, vol. 05, no. 02, pp. 441–457, 2009.

[199] J. Decraene, M. Chandramohan, M. Y. H. Low, and C. S. Choo, "Evolvable simulations applied to automated red teaming: A preliminary study," in *Simulation Conference (WSC), Proceedings of the 2010 Winter*, pp. 1444–1455, Dec 2010.

[200] R. Bronfman-Nadas, N. Zincir-Heywood, and J. T. Jacobs, "An artificial arms race: Could it improve mobile malware detectors?," in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, pp. 1–8, June 2018.

[201] H. Holm, M. Karresand, A. Vidström, and E. Westring, "A survey of industrial control system testbeds," in *Secure IT Systems* (S. Buchegger and M. Dam, eds.), (Cham), pp. 11–26, Springer International Publishing, 2015.

[202] D. A. Richard Candell, Keith A. Stouffer, "A Cybersecurity Testbed for Industrial Control Systems," in *Proceedings of the 2014 Process Control and Safety Symposium*, 2014.

[203] T. McAvoy and N. Ye, "Base control for the Tennessee Eastman problem," *Computers Chemical Engineering*, vol. 18, no. 5, pp. 383 – 413, 1994.

[204] N. L. Ricker, "Decentralized control of the Tennessee Eastman Challenge Process," *Journal of Process Control*, vol. 6, no. 4, pp. 205 – 221, 1996.

[205] P. Lyman and C. Georgakis, "Plant-wide control of the Tennessee Eastman problem," *Computers Chemical Engineering*, vol. 19, no. 3, pp. 321 – 331, 1995.

[206] S. Yin, S. X. Ding, A. Haghani, H. Hao, and P. Zhang, "A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process," *Journal of Process Control*, vol. 22, no. 9, pp. 1567 – 1581, 2012.

[207] T. McEvoy and S. Wolthusen, *A Plant-Wide Industrial Process Control Security Problem*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[208] M. Krotofil and A. A. Cárdenas, *Resilience of Process Control Systems to Cyber-Physical Attacks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[209] B. Genge, C. Siaterlis, I. N. Fovino, and M. Masera, "A cyber-physical experimentation environment for the security analysis of networked industrial control systems," *Computers Electrical Engineering*, vol. 38, no. 5, pp. 1146 – 1161, 2012. Special issue on Recent Advances in Security and Privacy in Distributed Communications and Image processing.

[210] T. Larsson, K. Hestetun, E. Hovland, and S. Skogestad, "Self-Optimizing Control of a Large-Scale Plant: The Tennessee Eastman Process," *Industrial & Engineering Chemistry Research*, vol. 40, no. 22, pp. 4889–4901, 2001.

[211] A. Isakov and M. Krotofil, "GitHub Repository: Damn Vulnerable Chemical Process - Tennessee Eastman." Available: `https://github.com/satejnik/DVCP-TE`, 2015. Last Accessed: 03/20/2016.

[212] A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson, "Attack models and scenarios for networked control systems," in *Proceedings of the 1st International Conference on High Confidence Networked Systems*, HiCoNS '12, (New York, NY, USA), pp. 55–64, ACM, 2012.

[213] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-14-079-02) Siemens SIMATIC S7-1200 Vulnerabilities," tech. rep., Mar 2014. Last Accessed: 10/01/2019.

[214] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-16-161-01) Siemens SIMATIC S7-300 Denial-of-Service Vulnerability," tech. rep., Jun 2016. Last Accessed: 10/01/2019.

[215] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-15-146-01): Mitsubishi Electric MELSEC FX-Series Controllers Denial of Service." Available: `https://ics-cert.us-cert.gov/advisories/ICSA-15-146-01`, 2015. Last Accessed: 19/12/2018.

[216] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-13-036-02) Ecava IntegraXor ActiveX Buffer Overflow," tech. rep., Feb. 2013. Last Accessed: 19/12/2018.

[217] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-16-103-02) Siemens SCALANCE S613 Denial-of-Service Vulnerability," tech. rep., Apr 2016. Last Accessed: 10/01/2019.

[218] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-11-335-01): 7-Technologies Data Server Denial of Service," tech. rep., 2011. Last Accessed: 19/12/2018.

[219] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-13-289-01) Cisco ASA and FWSM Security Advisories," tech. rep., Oct. 2013. Last Accessed: 19/12/2018.

[220] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-13-338-01) Siemens SINAMICS S/G Authentication Bypass Vulnerability," tech. rep., Dec. 2013. Last Accessed: 19/12/2018.

[221] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-16-049-01) B+B SmartWorx VESP211 Authentication Bypass Vulnerability," tech. rep., Feb. 2016. Last Accessed: 19/12/2018.

[222] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-11-356-01) Siemens Simatic HMI Authentication Vulnerabilities," Technical Report, 2011. Last Accessed: 19/12/2018.

[223] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-11-173-01) ClearSCADA Remote Authentication Bypass," tech. rep., Aug. 2011. Last Accessed: 19/12/2018.

[224] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-15-160-01A) N-Tron 702W Hard-Coded SSH and HTTPS Encryption Keys (Update A)," tech. rep., June 2015. Last Accessed: 19/12/2018.

[225] ICS-CERT, The Department of Homeland Security, "Advisory (ICSA-16-063-01) Moxa ioLogik E2200 Series Weak Authentication Practices," tech. rep., June 2016. Last Accessed: 19/12/2018.

[226] Y.-L. Huang, A. A. Cárdenas, S. Amin, Z.-S. Lin, H.-Y. Tsai, and S. Sastry, "Understanding the physical and economic consequences of attacks on control systems," *International Journal of Critical Infrastructure Protection*, vol. 2, no. 3, pp. 73 – 83, 2009.

[227] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, *A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

[228] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Technical Report: TIK-Report 103, 2001.

[229] J. Oesterle and L. Amodeo, "Efficient Multi-objective Optimization Method for the Mixed-model-line Assembly Line Design Problem," *Procedia CIRP*, vol. 17, pp. 82 – 87, 2014. Variety Management in Manufacturing.

[230] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of Genetic Algorithms*, pp. 69–93, Morgan Kaufmann, 1991.

[231] N. Riquelme, C. V. Lücken, and B. Baran, "Performance metrics in multi-objective optimization," in *2015 Latin American Computing Conference (CLEI)*, pp. 1–11, Oct 2015.

[232] A. Fielder, E. Panaousis, P. Malacaria, C. Hankin, and F. Smeraldi, "Decision support approaches for cyber security investment," *Decision Support Systems*, vol. 86, pp. 13 – 23, 2016.

[233] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, pp. 173–195, June 2000.

[234] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 263–282, 2002.

[235] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[236] A. Toffolo and E. Benini, "Genetic diversity as an objective in multi-objective evolutionary algorithms," *Evol. Comput.*, vol. 11, pp. 151–167, May 2003.

[237] T. Wagner, N. Beume, and B. Naujoks, "Pareto-, aggregation-, and indicator-based methods in many-objective optimization," in *Evolutionary Multi-Criterion Optimization* (S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, eds.), (Berlin, Heidelberg), pp. 742–756, Springer Berlin Heidelberg, 2007.

[238] J. Knowles and D. Corne, "The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 1, pp. 98–105 Vol. 1, July 1999.

[239] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates, "Pesa-ii: Region-based selection in evolutionary multiobjective optimization," in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, (San Francisco, CA, USA), pp. 283–290, Morgan Kaufmann Publishers Inc., 2001.

[240] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pp. 82–87 vol.1, June 1994.

[241] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, pp. 712–731, Dec 2007.

[242] B. Naujoks, N. Beume, and M. Emmerich, "Multi-objective optimisation using s-metric selection: application to three-dimensional solution spaces," in *2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1282–1289 Vol. 2, Sep. 2005.

[243] J. Bader and E. Zitzler, "Hype: An algorithm for fast hypervolume-based many-objective optimization," *Evolutionary Computation*, vol. 19, no. 1, pp. 45–76, 2011.

[244] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, April 2002.

[245] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, pp. 221–248, Sept. 1994.

[246] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 257–271, Nov 1999.

[247] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall, 1986.

[248] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[249] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

[250] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015.

[251] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," 1996.

[252] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–, Oct. 2017.

[253] S. Mahadevan, N. Marchalleck, T. K. Das, and A. Gosavi, "Self-improving factory simulation using continuous-time average-reward reinforcement learning," in *Proceedings of the 14th International Conference on Machine Learning*, pp. 202–210, Morgan Kaufmann, 1997.

[254] J. Kober, J. A. D. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, July 2013.

[255] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck, "Learning intrusion detection: Supervised or unsupervised?," in *Image Analysis and Processing – ICIAP 2005* (F. Roli and S. Vitulano, eds.), (Berlin, Heidelberg), pp. 50–57, Springer Berlin Heidelberg, 2005.

[256] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, California: Wadsworth, 1984.

[257] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[258] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[259] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, Mar 1986.

[260] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[261] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, p. 169–198, Aug 1999.

[262] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, p. 123–140, Aug. 1996.

[263] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.

[264] A. Geron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st ed., 2017.

[265] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119 – 139, 1997.

[266] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, (New York, NY, USA), pp. 144–152, ACM, 1992.

[267] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, Sep 1995.

[268] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, pp. 415–425, March 2002.

[269] N. Cristianini, *An introduction to Support Vector Machines : and other kernel-based learning methods / Nello Cristianini and John Shawe-Taylor.* Cambridge: Cambridge University Press, 2000.

[270] D. Barber, *Bayesian Reasoning and Machine Learning*. New York, NY, USA: Cambridge University Press, 2012.

[271] R. C. Berwick, "MIT Lecture Notes: An Idiot's guide to Support vector machines (SVMs)." Available: `http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf`, 2003. Last Accessed: 14/05/2019.

[272] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, pp. 1443–1471, July 2001.

[273] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, July 2006.

[274] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, Dec 1943.

[275] BrainFactsOrg, "The Neuron." Available: `https://www.brainfacts.org/brain-anatomy-and-function/anatomy/2012/the-neuron`, April 2012. Last Accessed: 13/04/2019.

[276] V. K. Ojha, A. Abraham, and V. Snášel, "Metaheuristic design of feedforward neural networks: A review of two decades of research," *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 97 – 116, 2017.

[277] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: Foundations of Research," ch. Learning Representations by Back-propagating Errors, pp. 696–699, Cambridge, MA, USA: MIT Press, 1988.

[278] A. Ng, "Machine Learning: Gradient Descent Intuition Lecture," online machine learning coursera module, 2017. Available: `https://www.coursera.org/`.

[279] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[280] L. N. Smith, "No More Pesky Learning Rate Guessing Games," *CoRR*, vol. abs/1506.01186, 2015.

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

[282] R. Pascanu, T. Mikolov, and Y. Bengio, "Understanding the exploding gradient problem," *CoRR*, vol. abs/1211.5063, 2012.

[283] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[284] C. Olah, "Understanding LSTM Networks." Available: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015.

[285] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[286] A. Ng, "Sparse Autoencoder," Lecture Notes, 2011. Available: `http://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf`.

[287] D. P. Kingma and M. Welling, "Auto-encoding variational bayes." arXiv, 2013.

[288] N. Ricker, "Optimal steady-state operation of the Tennessee Eastman challenge process," *Computers Chemical Engineering*, vol. 19, no. 9, pp. 949 – 959, 1995.

[289] C. Perone, "Pyevolve: A python open-source framework for genetic algorithms," *SIGEVOlution*, vol. 4, pp. 12–20, 11 2009.

[290] K. Mrugala, N. Tuptuk, and S. Hailes, "Evolving attackers against wireless sensor networks using genetic programming," *IET Wireless Sensor Systems*, vol. 7, no. 4, pp. 113–122, 2017.

[291] S. Luke, "Ecj then and now," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, (New York, NY, USA), pp. 1223–1230, ACM, 2017.

[292] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, "Deap: Evolutionary algorithms made easy," *J. Mach. Learn. Res.*, vol. 13, pp. 2171–2175, July 2012.

[293] B. L. Miller and D. E. Goldberg, "Genetic algorithms, selection schemes, and the varying effects of noise," *Evol. Comput.*, vol. 4, pp. 113–131, June 1996.

[294] J. Byron and W. Iba, "Population diversity as a selection factor: Improving fitness by increasing diversity," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, GECCO '16 Companion, (New York, NY, USA), pp. 953–959, ACM, 2016.

[295] D. Gupta and S. Ghafir, "An overview of methods maintaining diversity in genetic algorithms," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, pp. 56–60, 2012.

[296] 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 ReliabilitySpecial Issue - Genetic Algorithms and Reliability.

[297] K. Deb, *Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction*, pp. 3–34. London: Springer London, 2011.

[298] E. Zitzler, M. Laumanns, L. Thiele, C. M. Fonseca, and V. G. da Fonseca, "Why quality assessment of multiobjective optimizers is difficult," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, GECCO'02, (San Francisco, CA, USA), pp. 666–674, Morgan Kaufmann Publishers Inc., 2002.

[299] G. G. Yen and Z. He, "Performance metric ensemble for multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 18, pp. 131–144, Feb 2014.

[300] S. Wessing, "Implementations: Hypervolume." `https://ls11-www.cs.tu-dortmund.de/rudolph/hypervolume/start`. Last Accessed: 22/10/2018.

[301] M. Fleischer, "The measure of pareto optima applications to multi-objective metaheuristics," in *Evolutionary Multi-Criterion Optimization* (C. M. Fonseca, P. J. Fleming, E. Zitzler, L. Thiele, and K. Deb, eds.), (Berlin, Heidelberg), pp. 519–533, Springer Berlin Heidelberg, 2003.

[302] C. M. Fonseca, L. Paquete, and M. Lopez-Ibanez, "An improved dimension-sweep algorithm for the hypervolume indicator," in *2006 IEEE International Conference on Evolutionary Computation*, pp. 1157–1163, July 2006.

[303] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – a comprehensive introduction," *Natural Computing*, vol. 1, pp. 3–52, Mar 2002.

[304] E. Byres, "Revealing network threats, fears: How to use ANSI/ISA-99 standards to improve control system security." Available: `https://www.isa.org/link/networkthreats/`, 2011.

[305] B. Iglewicz and D. C. Hoaglin, *How to detect and handle outliers*. ASQC Quality Press, 1993.

[306] T. G. Dietterich, "Machine learning for sequential data: A review," in *Structural, Syntactic, and Statistical Pattern Recognition* (T. Caelli, A. Amin,

R. P. W. Duin, D. de Ridder, and M. Kamel, eds.), (Berlin, Heidelberg), pp. 15–30, Springer Berlin Heidelberg, 2002.

[307] W. Hu, W. Hu, and S. Maybank, "AdaBoost-Based Algorithm for Network Intrusion Detection," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, pp. 577–583, April 2008.

[308] T. Friedrich and M. Wagner, "Seeding the initial population of multi-objective evolutionary algorithms: A computational study," *Applied Soft Computing*, vol. 33, pp. 223 – 230, 2015.

[309] Y. Machida, S. Ootakara, H. Seki, *et al.*, "Vinyl Acetate Monomer (VAM) Plant Model: A New Benchmark Problem for Control and Operation Study," *IFAC-PapersOnLine*, vol. 49, no. 7, pp. 533 – 538, 2016. 11th IFAC Symposium on Dynamics and Control of Process SystemsIncluding Biosystems DYCOPS-CAB 2016.

[310] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population based training of neural networks," *CoRR*, vol. abs/1711.09846, 2017.