UNIVERSITY OF WALES TRINITY SAINT DAVID

Bacterial Cyber Security

Smart Moving Target Defense Systems

by

Nathan Hughes

A project proposal submitted in partial fulfillment for the degree of M.Sc. Applied Computing

in the
School of Applied Computing
Faculty of Architecture, Computing and Engineering

Supervisor: Mabrouka Abuhmida

January 2020

Declaration of Authorship

I, Nathan Hughes declare that I am the sole author of this project proposal; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed: ----------

Date: ----------

Form of Consent

I Nathan Hughes hereby consent that my Project, submitted in candidature for the M.Sc. Applied Computing degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed: ----------

Date: ----------

UNIVERSITY OF WALES TRINITY SAINT DAVID

Abstract

Faculty of Architecture, Computing and Engineering
School of Applied Computing

M.Sc. Applied Computing

by Nathan Hughes

Moving Target Defence is a new field of cyber security that seeks to disrupt the asymmetry of current attack-defence scenarios on static systems, achieved by changing the attack surface so that vulnerabilities are short-lived and potentially rendered unusable. Theoretical models of these defences exist but do not suitably address the efficiency of the system's transitions. In this thesis, a continuous-time, fixed-order Markov model of Moving Target Defence systems is created that allows for analysis of both security and efficiency of state transitions.

Based on theoretical statistical approximations of the measured simulation properties, a multi-objective genetic algorithm is employed to search for optimal solutions in the solution space, given a set of states and probabilistically-generated vulnerabilities. The structure of the solution sets are comparatively analysed by order of the Markov chain, and found to have significant effect on both the security and efficiency of the systems. Based on these results, further research questions are posed on how the model can be developed to account for more complex behaviour from both the defender and potential adversaries.

Contents

Declaration of Authorship i

Form of Consent ii

Abstract iii

List of Figures vi

List of Tables vii

Symbols viii

1 Introduction 1
- 1.1 Static Defences and the Rise of Exploit Kits . . . 1
- 1.2 Moving Target Defence as a Solution . . . 2
- 1.3 Thesis Objectives . . . 3

2 Literature Review 4
- 2.1 Probabilistic Models of Moving Target Defences . . . 4
  - 2.1.1 A Game Theoretic Approach to Strategy Determination for Dynamic Platform Defenses [1] . . . 4
  - 2.1.2 Markov Modelling of Moving Target Defence Games [2] . . . 7
  - 2.1.3 Empirical Game-Theoretic Analysis for Moving Target Defense [3] 8
- 2.2 Evolutionary Models for Moving Target Defence . . . 8

3 Methodology 10

4 Modelling and Theoretical Results 11
- 4.1 Limitations of Previous Work . . . 11
- 4.2 Graphs and Stochastic Processes . . . 12
- 4.3 Historical Digraph . . . 13
- 4.4 Random Walk on Historised Graph . . . 16

5 Implementation and Design 19
- 5.1 Data Structure . . . 19

5.2 Representation of an Adversary . . . 20
5.3 Simulation . . . 20
5.3.1 Combined Fitness Function . . . 21
5.4 Deterministic Approximation of Simulation . . . 22
5.4.1 Approximating m^{sim} . . . 22
5.4.2 Approximating t_a . . . 23
5.4.3 Approximating t_a^{con} . . . 25
5.5 Pareto Optimisation . . . 26
5.6 Multi-objective Genetic Algorithm . . . 28
5.6.1 Outline of Algorithm . . . 28
5.7 Comparison to Random Samples . . . 31

6 Data Analysis and Results **32**
6.1 Comparison of Pareto Fronts for different values of k . . . 32
6.2 Correlation of f_2 and f_3 . . . 35
6.3 Relationship between Standard Deviation of f_1 and k . . . 36
6.4 Discussion . . . 36

7 Conclusion and Further Work **37**
7.1 Conclusion . . . 37
7.2 Further Work . . . 38
7.2.1 Theoretical Development of Front Geometry . . . 38
7.2.2 Development of Sophisticated Adversary Models . . . 38
7.2.3 Varying values of k . . . 38
7.2.4 Layered Moving Target Defence . . . 39
7.2.5 Nash Equilibria . . . 39

A Cullen and Frey graphs **40**

Bibliography **43**

List of Figures

4.1 On the left, a complete graph on $n = 4$ states. On the right, a historised graph on the same $n = 4$ states with history length $k = 2$ 14

5.1 An example dictionary $G^{rep}_{4,2}$ representing a 2-avoiding MTD system on 4 states. 20

5.2 Plot of the mean number of transitions of 10 simulations run for 500 units of time versus the approximation in terms of the stationary distribution of 1000 random $6, 3$-MTD graphs, where $|\mathcal{A}| = 10$ is randomly generated. The Pearson correlation of this set of data is 0.955. 23

5.3 Plot of the mean time spent compromised of 10 simulations run for 500 units of time versus the Perron compromised circulation of 1000 random $6, 3$-MTD graphs, where $|\mathcal{A}| = 10$ is randomly generated. The Pearson correlation is 0.961. 24

5.4 Plot of the mean time compromised per bug of 10 simulations run for 500 units of time versus the approximation using equations 5.7 and 5.8 for 1000 $6, 3$-MTD graphs with sensitivity 0.001, where $|\mathcal{A}| = 10$ is randomly generated. The Pearson correlation of this set of data is 0.977. 26

5.5 Combined plot of Pareto fronts for $n = 4, k \in \{1, 2\}$, represented by red circles and blue crosses respectively, and 100 random $4, 1$- and $4, 2$-graphs, represented by green squares and yellow diamonds. Observe that most, if not all, green squares and yellow diamonds are dominated by the red circles and blue crosses. The single adversary used compromised the states (1,2). 31

6.1 Combined plot of Pareto fronts for $n = 6, k \in \{1, 2, 3, 4\}$, represented by red circles, blue crosses, green squares and yellow diamonds respectively. 33

6.2 Combined plots of value density of fitness function components f_1, f_2, f_3 for $n = 6, k \in \{1, 2, 3, 4\}$, represented by red, blue, green and yellow curves respectively. 34

6.3 Plot of (f_2, f_3) components of all Pareto fronts (blue) against the approximation with equation 6.1 (red). 36

A.1 Cullen and Frey plots of distribution of f_1 values for $n = 6, k \in \{1, 2, 3, 4\}$ with 500 accompanying bootstraps. 40

A.2 Cullen and Frey plots of distribution of f_2 values for $n = 6, k \in \{1, 2, 3, 4\}$ with 500 accompanying bootstraps. 41

A.3 Cullen and Frey plots of distribution of f_3 values for $n = 6, k \in \{1, 2, 3, 4\}$ with 500 accompanying bootstraps. 42

List of Tables

6.1 Statistical properties of first fitness component of members of the Pareto front by value of k 35
6.2 Statistical properties of second fitness component of members of the Pareto front by value of k 35
6.3 Statistical properties of third fitness component of members of the Pareto front by value of k 35

Symbols

$\mathcal{S}$	state space
n	cardinality of $\mathcal{S}$, the number of states
k	length of history state
$V(G)$	vertex set of the graph G
$E(G)$	edge set of the graph G
$\mathcal{H}_{n,k}$	set of all k-histories on n states
$\mathcal{E}_{n,k}$	set of all pairs of k-histories on n states satisfying compatibility criterion
$G_{n,k}$	directed graph with vertex set $\mathcal{H}_{n,k}$ and edge set $\mathcal{E}_{n,k}$
W	weight matrix associated to the weighted directed graph $G_{n,k}$
P	probability transition matrix
$\mathbb{P}(A)$	the probability of event A occurring
$\mathbb{E}[X]$	expected value of the random variable X
π, ϕ	stationary distribution of a random walk

Chapter 1

Introduction

1.1 Static Defences and the Rise of Exploit Kits

Exploit kits are tools that automate the deployment of a collection of exploits, targeting the infection vectors of said exploits to install malware in the targeted computers [4]. In 2013, the exploit kit *Angler* was released [5]. Amongst other activities, users of Angler were able to:

1. exploit vulnerabilities of Adobe Flash and Internet Explorer [5, 6],
2. injecting redirecting `POST` methods into legitimate web pages pointing to malicious servers [5],
3. employ a technique called *domain shadowing* in which the Angler user compromises the account of a domain name registrant and create sub-domains that point to malicious servers [5–7].

Angler was brought to an end on the 6th of October 2015 by Cisco, but not before the kit had generated up to $60 million of revenue for its users [6]. The implication of the existence of such software is clear - not only are they contributing to the cyber crime industry, but of particular note is the automatic nature of such kits. More recent research has shown that the sophistication of these kits has grown rapidly, to a point where the automation of the delivery of malware is now suited for mass proliferation through advertisements on web-pages, known as *malvertising* [8].

These exploits are able to be automated because the exploits are able to be re-used between systems. Often, these exploits remain hidden to the developers. It is unlikely

that any developments in computer security will ever address the existence of zero-day vulnerabilities directly. It is imperative, therefore, that these vulnerabilities are addressed indirectly.

1.2 Moving Target Defence as a Solution

A system implementing a Moving Target Defence technique is a system that makes it's attack surface less static and homogeneous, giving the defender the same asymmetric advantage over time previously only granted to the attacker [9, 10].

A survey by Ward *et al.* taxonomically defined five moving target techniques that defenders could implement in their systems [11]. These are as follows:

Dynamic Data which changes the structure of data used and offered by the application.

Dynamic Software which changes the code in the application.

Dynamic Runtime Environment which changes the environment offered by the operating system. Ward *et al.* sub-categorised this further, making a distinction between Address Space Randomization, which changes the layout of memory, and Instruction Set Randomization, which changes the interface offered by the operating system.

Dynamic Platforms which changes the properties of the platform on which the application runs, including but restricted to operating system version and architecture of the CPU.

Dynamic Networks which changes, for example, the protocols and addresses of the network.

As early as 2009, researchers have recognised Moving Target Defence as a realistic strategy for defence, achievable with the innovations in technology available at the time - key technologies include, but are not limited to, virtualisation, address space layout randomisation and Software as a Service offered by cloud providers [10]. Further research has recognised MTD defence strategies as an effective strategy to prevent DDoS attacks [12].

It is important to note that the vulnerabilities of the technologies incorporated into a Moving Target Defence system are still present, however these vulnerabilities are less to exploited by attackers for long due to the evolving nature of the system.

1.3 Thesis Objectives

The research field of Moving Target Defence systems has lacked analysis of system efficiency. The aim of this thesis is to introduce a model that allows the analysis of both security and efficiency to be conducted simultaneously. Such analysis has yet to be conducted mainly due to the discrete-time nature of most models. For example, probabilistic analysis of a Moving Target Defence system in discrete time has shown that avoiding previously-chosen states secures the system from both static and adaptive adversaries [1]. Another model bounds the rate of IP address transition against the rate an adversary attempts a known set of IPs [2].

During the last decade, it was discovered that approximately 50% of bacteria adaptively defend themselves against viruses using an immune system called CRISPR-CAS [13]. Such a system uses parts of the infecting phage to adapt themselves against the attack. The aim of this project is to begin building a model that is a Moving Target Defence analogue of this system: rather than directly neutralising such a threat with patches and bugfixes by developers, adapting the system's evolution to neutralise the threat indirectly.

The model presented in this thesis will generalise the properties of the research mentioned above to continuous-time, creating a self-avoiding stochastic process on the provided states, with transition rate dependant on both the current state and the recent history of states. To find optimal solutions for the associated fitness functions describing the desired properties, a Multi-Objective Evolutionary Algorithm is used to find the set of non-dominating solutions. The properties of these models are analysed statistically for significant differences in efficiency of state transfer and security against static adversaries generated according to state similarity.

Chapter 2

Literature Review

This chapter discusses the current literature surrounding the theory of Moving Target Defence systems. Due to the wide range of research produced on this topic in recent years, particular emphasis is given to the mathematical frameworks used and the outcomes of any evolution-based experiments. This literature review is divided into two sections - one for probabilistic modelling and another for application of evolutionary techniques. The involved nature of the mathematical models warrants a subsection for each paper, so that the concepts can be given proper treatment.

2.1 Probabilistic Models of Moving Target Defences

2.1.1 A Game Theoretic Approach to Strategy Determination for Dynamic Platform Defenses [1]

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a finite collection of platforms through which the MTD system migrates. Associated with these platforms is a boolean $v(p_i)$ which denotes the vulnerability of the platform to a zero-day exploit.

This paper seeks optimal strategies for the defender of a MTD system. The authors consider two scenarios that differ by the attacker's goal. In the first scenario, known as 'Crash the Satellite', the attacker tries to control the system for some amount of time T to win. In the second scenario, known as 'Exfiltration Over a Difficult Channel', after controlling the system for T time the attacker's pay-off function begins to increase.

This paper presents the hacking scenario as an alternating two-player game with incomplete and imperfect information. Specifically, the attacker is aware of all moves made by the defender but not the full range of moves available (perfect and incomplete), whereas

the defender has no information as to which attacks have been carried out and which vulnerabilities the attacker can exploit (imperfect and incomplete). The defender is the leader of the game, who chooses a platform $p^k \in P$ on turn k. The attacker follows by playing an exploit if and only if $v(p_k)$ is true. The aim of the defender, therefore, is to choose a sequence of platforms such that the attacker cannot compromise the system for T time.

Of particular interest in this model is the measure of statistical variation between platforms. Such a measure can be based on the similarity of modules or code between platforms, for example, and is used as an indicator of how likely an exploit can be used for both platforms.

Two types of attackers are considered. The first is a static adversary, that is not able to develop exploits during the game's lifetime. As such, the defender only wishes to minimise the probability of continuous compromise, governed by equation 2.1.

$$\mathbb{P}(C(k)) := \mathbb{P}\left(\bigcap_{i=k}^{k-K+1} v(p^i)\right) \tag{2.1}$$

where K is an integer corresponding to the number of previous platforms required to meet the time criterion T. Using the identity $\mathbb{P}(A \cap B) = \mathbb{P}(A|B)\mathbb{P}(B)$ and approximating with via the conditional independence equation $\mathbb{P}(A|B \cap C) = \mathbb{P}(A|B)\mathbb{P}(A|C)$, the defender's best next move is determined by equation 2.2.

$$p^k = \arg\min_j \mathbb{P}\left(v(p_j)| \bigcap_{i=k-K+1}^{k-1} v(p^i)\right) \tag{2.2}$$

The second variation of attacker is adaptive, which more suitable for modelling long-term games. This attacker can develop an exploit to a platform that is presented by the attacker, determined by the length of time they are exposed to said platform. As such, the defender must not only minimise the probability of compromise but also of the attacker developing an exploit. This corresponds to equation 2.3.

$$p^k = \arg\min_{p_j} \lambda\mathbb{P}(C(k)) + (1-\lambda)\mathbb{P}(E(k)) \tag{2.3}$$

where $E(k)$ is the boolean representing an exploit being developed for platform p_k at time step k and $0 \leq \lambda \leq 1$ is a parameter weighing both actions relative to their dependent successes: if it is very likely that the system is exploitable, then there is very likely an exploit for said system, so $\mathbb{P}(E(k))$ is small, and vice versa.

To explicitly define $\mathbb{P}(E(k))$, the authors consider an arbitrary probability density function f_p that governs the hacker's chances of developing an exploit for platform p.

$$\mathbb{P}(E(k)) = f_{p^k}(t(p^k))(1 - \mathbb{P}(v(p^k))) \tag{2.4}$$

Given that $t(p)$ observations of p have been made, the probability that the attacker has developed an exploit is simply $F_p(t(p))$, where F_p is the cumulative distribution function associated to f_p, which yields

$$\mathbb{P}(v(p^k)) = 1 - \prod_{j=1}^{K-1} \left(1 - F_{p^{k-j}}\left(t(p^{k-j})\right)\mathbb{P}\left(v(p^k)\middle|v(p^{k-j})\right)\right) \tag{2.5}$$

Using 2.5, and using via conditional independence in a similar fashion to 2.2, one can obtain an approximation for $\mathbb{P}(v(p_i)| \cap_{j=1}^{K-1} v(p^{k-j}))$.

These findings were simulated against other strategies, such as uniform selection of the platforms, with both static and adapting adversaries. Five platforms were used, where one platform was defined as vulnerable to exploit and the rest had a probability of exploit equal to the code similarity between them and the vulnerable platform. The attacker's goal was to compromise the system for $K = 3$ steps.

In the static Crash the Satellite case, only 2% of trials did the derived strategy using 2.2 result in a failed defence, compared to 79% of trials where each platform was randomly selected. In the Exfiltration over a Difficult Channel scenario, the derived strategy protected the system successfully in 98% of the trials, whereas the uniform case averaged a 26% compromise rate.

In the adaptive case, the strategy that uses 2.2 was simulated along with the optimised strategy 2.3 and uniform selection. The author's assumed the probability density function governing the development an exploit for a platform p followed a Gamma$(2, 25)$ distribution:

$$f_p(t) = \frac{t}{625}e^{-\frac{t}{25}}$$

In the Exfiltration over a Difficult Channel scenario, 94% of trials saw no compromise when following 2.2 or 2.3, whereas only 49% of trials were as successful with the uniform strategy.

2.1.2 Markov Modelling of Moving Target Defence Games [2]

In this paper, a MTD scenario is modelled with a Markov chain. A more specific subclass of MTD games is described in detail, in which the attacker and defender have a rate of play λ and μ respectively. Supposing that the game has k states, the associated probability transition matrices M^A, M^D governing the probability of state transition for the attacker and defender respectively. Moves are decided based on uniform selection of either the defender, attacker or neither. If selected, the player can take a turn.

The authors provide an explicit example of this model by considering an IP hopping defence. Suppose there are n IPs that the MTD system can choose. The defender begins at state 0 and their goal os to reach state, corresponding to the defender successfully choosing the correct IP. If the attacker takes a turn and guesses correctly, they win the game. If the attacker guesses the incorrect IP address, they proceed to the state above. If the defender takes a move then the attacker is reset to sate 0.

A scenario where the system hides multiple targets is also considered. Suppose, for example, that there are n possible IPs which hide k targets used by the system. The defender wishes to discover $k' < k$ IPs to win. When the defender moves, they select one of the k platforms and reassigns it to one of the other $n - k$ unused IPs. The attacker, on the other hand, is adaptive in the sense that if a target they knew the location of is moved by the defender then they will know that it has moved. The Markov chain corresponding to this is a modified discrete-time birth-death process.

The first theorem of this paper concerns the probability of the attacker successfully winning the game in an allotted time.

Theorem 2.1. *Suppose the MTD game has transition matrix M with stationary distribution π such that $\pi M = \pi$. The probability that the attacker wins in th first T turns is bounded above by $T\pi(\mathbf{WIN})$, where* **WIN** *is the winning state.*

The paper also considers the cost associated with attacking a MTD system successfully, discussed in Theorem 2.2.

Theorem 2.2. *Suppose α is the cost associated with an attacker's move during a time step. The probability that the attacker wins by spending less than C is bounded above by $\frac{C}{\alpha}\pi(\mathbf{WIN})$.*

An important concept introduced in this paper is the security capacity of the MTD game, defined by $c = -\log(\pi(\mathbf{WIN}))$. Intuitively, for any s, t that sum to c, the probability that the attacker wins in the first 2^t turns is bounded above by 2^{-s}.

In the case of single target IP hopping, the security capacity is can be approximated by $\log(\frac{\lambda n}{\mu})$. For multiple target hiding, the security capacity is at least $k' \log(\frac{\lambda(n-k'+1)}{\mu(k-k'+1)})$.

Finally, the layered composition of MTD is considered. A sequence $M_0, M_1, \ldots M_f$ of games are played simultaneously by the defender. The attacker begins by playing game M_0, and can only begin playing game M_j if they are in the winning position of game M_{j-1}. If λ_j, μ_j are the corresponding rates of play for the defender and attacker associated to game M_j, then the security capacity of the layered MTD game as a whole is at least

$$\sum_{j=0}^{f} \Big(c_j + \log \lambda_j \Big) \tag{2.6}$$

2.1.3 Empirical Game-Theoretic Analysis for Moving Target Defense [3]

This paper models a game with finitely-many states, each of which can be controlled by either the attacker or defender. The paper discusses strategies that each player can employ to control the system using the theory of Nash Equilibria. A total of 72 game instances were discovered, some of which were dynamic Nash equilibria, such as

Maximal Defence - the defender has such an aversion to attacks that the attacker is unable to compromise enough of the system, so gives up,

Maximal Attack - the attacker is so successful at attacking states that the defender cannot do any better than to give up,

Fight - the attacker and defender are both successful at attacking and defending respectively, resulting in constant tension over each state and no win condition being achieved by either party,

Share - the attacker and defender achieve their win conditions simultaneously and no longer attempt to take states from the other party.

2.2 Evolutionary Models for Moving Target Defence

The concept of evolution has been applied to Moving Target Defence systems by several authors. One paper discusses two genetic algorithms for producing a Moving Target Defence system, differing by mutation process [14]. The first algorithm mutates the

values of the parameters, whilst the other will mutation the domain of a parameter. Each algorithm consists of 140 chromosomes in each generation, where each chromosome is a composition of 102 parameters. Selection fo the mutations was done uniformly with 25% probability and via two-round tournament selection with 75% probability. The authors recommend further work to this selection process as an improvement of the model. Another paper has applied the theory of Moving Target Defence to Software-defined Networks, arguing that their large search space makes them an ideal candidate for application [15]. A thesis from 2014 uses a genetic algorithm to evolve a set of system configurations according to attacks that act as predators [16]. A prototype was developed that could withstand a small volume of attacks and adapt accordingly.

Chapter 3

Methodology

In chapter 4 of this thesis, a model is developed of a Moving Target Defence system. The statistical properties of the model are proved using common properties of abstract graphs and well-known mathematical theorems from graph theory and probability theory. This is done so that the long-term behaviour of these processes can be understood and assigned numeric values without repeated simulation.

In chapter 5, a realisation of this model is detailed with a view towards implementation in Python 3.7. The objective functions of the optimisation problem are identified and, using the properties proven in the previous chapter, have their long-time limit approximated. Such approximations are done so that there is no need to run direct simulations for a long time to ensure accurate results.

Chapter 6 deals with the statistical analysis of the data obtained from a typical run of the genetic algorithm. The number of states was kept constant for all runs of the algorithm, as was the set of adversaries, to reduce the number of variables affecting the results. The analysis was conducted with libraries in both R and Python.

Chapter 4

Modelling and Theoretical Results

4.1 Limitations of Previous Work

Some models of this movement were presented in chapter 2 but all authors recognise that the discrete-time nature of their models restricted the insight they could provide [1, 2]. The main body of this thesis is the construction of a continuous-time model that appropriately adapts the results of these works to a real-time system.

The model constructed in this thesis is graph-based. The states of the system are modelled by vertices and the connecting edges describe the possible movement between states. To form a continuous-time model, a probability transition matrix and a holding-time vector are associated to the graph.

The nature of the main research question of this thesis is to do with the interplay of security and efficiency of any given pair of probability transition matrix and holding-time vector and how previous research effects the results. The goal is to obtain an optimal pair that minimises the number of transitions whilst effectively securing the system from compromise.

One might attempt to construct a continuous-time stochastic process on a complete graph, K_n, where n is the number of states. Previous work advises against randomly choosing subsequent states as this can create a system that is easily compromised [1]. Applying the suggested selection technique - by avoiding selecting a state preciously chosen in the last k steps for some k - one encounters a time-inhomogeneous stochastic process. This thesis presents analysis of such a process.

4.2 Graphs and Stochastic Processes

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and let $(\mathcal{S}, \mathcal{P}(\mathcal{S}))$ be the state space endowed with the power set σ-algebra. Let $I \subset \mathbb{R}$ be a set of infinite cardinality. Then $X : I \times \Omega \to \mathcal{S}$, defined by $X(i, \omega) = X_i(\omega)$ is a *stochastic process* indexed by I, with shortened notation $X = \{X_i : i \in I\}$. The random variables X_i are measurable in the sense that for any $i \in I$, $X^{-1}(S) = \{\omega \in \Omega : X(\omega) \in \mathcal{P}(\mathcal{S})\} \in \mathcal{F}$. An element $\omega \in \Omega$ is called a *sample path*. For brevity, the notation $X = X(\omega)$ is used, so that $\mathbb{P}(X \in F) := \mathbb{P}(\{\omega \in \Omega : X(\omega) \in F\})$ for any $F \in \mathcal{F}$.

A discrete-time stochastic process X on $\mathcal{S}$ is a *Markov chain* if it satisfies the Markov property:

$$\mathbb{P}(X_{n+1} = s_{n+1} | X_n = s_n, \ldots, X_0 = s_0) = \mathbb{P}(X_{n+1} = s_{n+1} | X_n = s_n). \tag{4.1}$$

That is, the probability of transferring to any state at time step $n + 1$ depends only on the state of X at time step n. This is also known as the *memoryless* property. A Markov chain X is *time homogeneous* if

$$\forall n \Big(\mathbb{P}(X_{n+1} = s_{n+1} | X_n = s_n) = \mathbb{P}(X_n = s_n | X_{n-1} = s_{n-1}) \Big). \tag{4.2}$$

The model constructed in this paper is a stochastic process on the space of possible states of a Moving Target Defence systems. To realise these processes, it will be useful to consider their state transition diagram. Let X be such a Markov chain on a state space $\mathcal{S}$ in discrete time (indexed by $\mathbb{N}$, say). To X, we may associate a directed graph (or digraph) $G(X)$ where

$$V\big(G(X)\big) = \mathcal{S},$$
$$E\big(G(X)\big) = \Big\{ (u, v) : u, v \in \mathcal{S} \wedge \forall n \in \mathbb{N} \big(\mathbb{P}(X_{n+1} = v | X_n = u) > 0 \big) \Big\}.$$

That is, the digraph has vertices in bijection with the states of X and a directed edge connects two states if the one-step transition probability is non-zero.

To construct a continuous-time Markov chain, also known as a Markov process, Y from a discrete-time Markov chain X, consider the probability transition matrix P of X and a vector of exponentially distributed random variables $T = (T_s)_{s \in \mathcal{S}}$, known as the *holding times*. These can be combined to form a transition rate matrix Q in the following manner:

$$Q_{ij} = \begin{cases} -\frac{1}{\mathbb{E}[T_i]} & i = j, \\ \frac{P_{ij}}{\mathbb{E}[T_i]} & i \neq j. \end{cases} \tag{4.3}$$

4.3 Historical Digraph

Let $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ be the set of all possible states for a Moving Target Defence system. Assume that $n \geq 3$.

Definition 4.1. Let $k < n$. A $k-$history at a state s_i is a partial permutation $(s_{i_1}, s_{i_2}, \ldots s_{i_{k-1}}, s_{i_k})$ of $\mathcal{S}$. The more explicit notation $(s_{i_1}, s_{i_2}, \ldots s_{i_{k-1}} | s_{i_k})$ is used, where s_{i_k} is the current state and the previous states are read in reverse order, $s_{i_{k-1}}, \ldots, s_{i_1}$.

The history at a state contains information about which states were previously visited in which order. The current state is stored as the final component. An example digraph is displayed in figure 4.1.

Definition 4.2. The k-history $g = (g_1, \ldots | g_k)$ is compatible with $h = (h_1, \ldots | h_k)$, writing $g \to h$, if $g_{l+1} = h_l$ for $l = 1, \ldots, k-1$ and $g_1 \neq h_k$.

Each k-history can also be interpreted as a self avoiding walk on the complete graph K_n of length k. The structure of the space of self-avoiding walks has been studied previously [18, 19].

Example 4.1. *Let $\mathcal{S} = \{1, 2, 3, 4\}$, and let $k = 3$. The history $h_1 = (1, 2|3)$ is compatible with the history $h_2 = (2, 3|4)$, but not compatible with $h_3 = (2, 3|1)$.*

The compatibility criterion describes the states previously visited which are to be avoided. This describes an irreflexive, asymmetric and intransitive relation on the set of all self-avoiding walks of length k on the complete graph K_n.

It is easy to see that for a state space $|\mathcal{S}| = n$, with history length k, there are a total of $n(n-1)\cdots(n-k+1) = \frac{n!}{(n-k)!}$ possible histories. Let $\mathcal{H}_{n,k}$ be the set of all such histories, and consider a digraph $G_{n,k}$ with vertices corresponding to $h \in \mathcal{H}_{n,k}$ and edges generated by the compatibility criterion: a directed edge (g, h) connects g to h if $g \to h$. Let $\mathcal{E}_{n,k}$ be the set of all edges of $G_{n,k}$.

Proposition 4.3. *The digraph $G_{n,k} = (\mathcal{H}_{n,k}, \mathcal{E}_{n,k})$ is $(n-k)$-regular. That is, let h_i be a vertex of $G_{n,k}$. Then*

$$\big|\{(h_i, h_j) | h_j \in \mathcal{H}_{n,k}\}\big| = n - k = \big|\{(h_j, h_i) | h_j \in \mathcal{H}_{n,k}\}\big|. \tag{4.4}$$

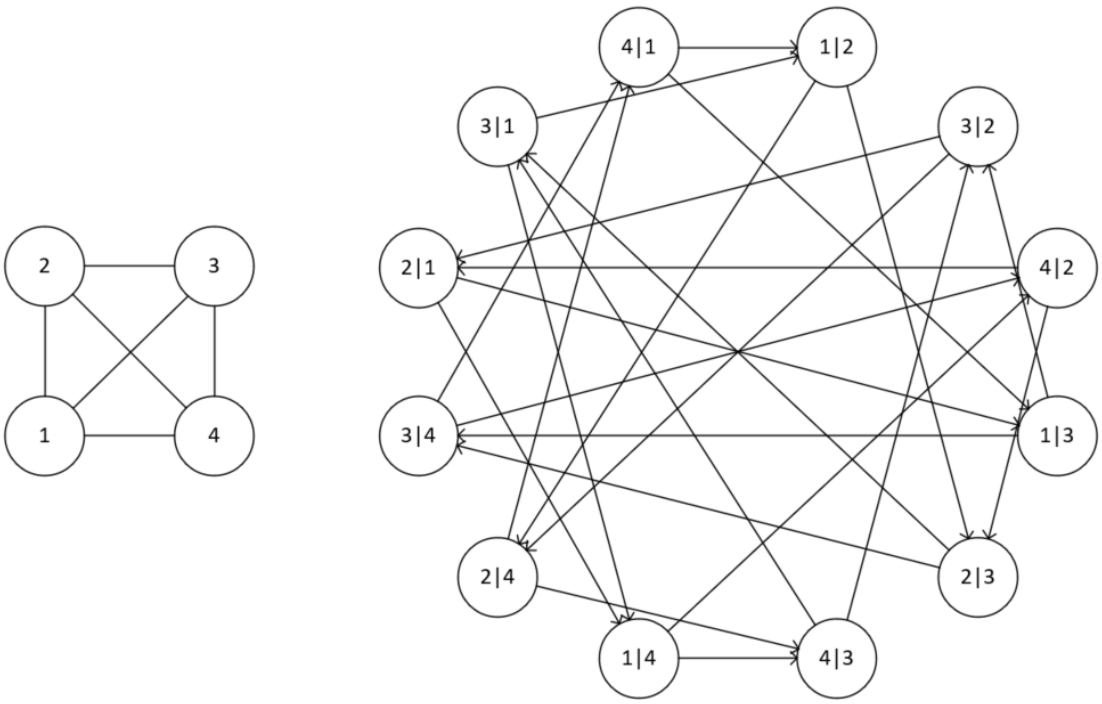


FIGURE 4.1: On the left, a complete graph on $n = 4$ states. On the right, a historised graph on the same $n = 4$ states with history length $k = 2$

Proof. Let $h_i = (h_{i_1}, h_{i_2}, \ldots | h_{i_k})$. Take $h_j \in \mathcal{H}_{n,k}$ with $h_i \to h_j$. Then h_j must take the form $(h_{i_2}, h_{i_3}, \ldots h_{i_k} | x)$ where $x \in \mathcal{S}$. There are $n - k$ possible choices for x that constructs a valid history, since $|\mathcal{S}| = n$ but we cannot select any of $h_{i_1}, \ldots h_{i_k}$.
Now suppose $h_j \to h_i$. Then $h_j = (y, h_{i_1}, \ldots | h_{i_{k-1}})$ for some $y \in \mathcal{S}$. Similarly, there are $n - k$ possible choices for y. □

Using this proposition, the number of edges can be calculated as$|\mathcal{E}_{n,k}| = |\mathcal{H}_{n,k}|(n-k) = \frac{n!}{(n-k-1)!}$.

Definition 4.4. A digraph G with vertex and edge sets $V(G), E(G)$ is *strongly connected* if there exists a path along edges of G between any two vertices. That is, for any $v, v' \in V(G)$ there exists a sequence of vertices $u_1, \ldots, u_n$ such that $(v, u_1), (u_n, v') \in E(G)$ and for all $1 \leq i \leq n - 1$, $(u_i, u_{i+1}) \in E(G)$.

Proposition 4.5. *Let $G_{n,k}$ be the digraph on $\mathcal{H}_{n,k}$. Then $G_{n,k}$ is strongly connected if and only if $k \leq n - 2$.*

Proof. Suppose $k = n-1$. By proposition 4.3, there is only one edge leaving each vertex. Let $h = (h_1, \ldots, h_i, h_{i+1}, \ldots, h_{n-1})$. We follow the only orbit that contains h.

$$
\begin{aligned}
h = (h_1, \dots | h_{n-1}) &\to \\
(h_2, \dots | h_n) &\to \\
&\vdots \\
(h_n, \dots | h_{n-2}) &\to \\
(h_1, \dots | h_{n-1}) &= h
\end{aligned}
$$

There are n histories in the orbit. Since $|\mathcal{H}_{n,n-1}| = n! \neq n$ for $n \geq 3$, $G_{n,k}$ cannot be connected, hence is not strongly connected.

Now let $k \leq n-2$. Let σ be a permutation on $\{1, 2, \dots, n\}$, and define $\sigma(h) = \sigma(h_1, h_2, \dots | h_k) := (\sigma(h_1), \sigma(h_2), \dots | \sigma(h_k))$. We want to find a path from h to $\sigma(h)$. To do this, we show that we can apply any transposition to h; that is, given any pair of symbols $s_i, s_j \in \mathcal{S}$, with $\sigma = (s_i, s_j)$, a path from h to $(s_i, s_j)(h)$ exists. From this, we deduce the existence of a path from h to $\sigma(h)$ for any $\sigma \in S_n$, since permutations can be constructed from cycles, which in turn are constructed from transpositions. An example decomposition of a permutation is $(a_1, a_2, \dots, a_k) = (a_1, a_2)(a_1, a_2) \cdots (a_1, a_k)$. The case that a transposition swaps two symbols not in h is trivial. If one symbol, h_i say, is present in h but h_j is not, we proceed as:

$$
\begin{aligned}
(h_1, \dots, h_i, \dots | h_k) &\to \\
(h_2, \dots, h_i, \dots | x) &\to \\
&\vdots \\
(h_i, \dots, h_k, \dots | h_{i-2}) &\to \\
(h_{i+1}, \dots, h_k, \dots | h_{i-1}) &\to \\
(h_{i+2}, \dots, h_k, \dots | h_j) &\to \\
&\vdots \\
(h_1, \dots, h_j, \dots | h_k)
\end{aligned}
\tag{4.5}
$$

To demonstrate the existence of transpositions that swap two elements present in g, we construct an algorithm for swapping adjacent symbols, and show that compositions of these operations can create any transposition. Let $h = (h_1, \dots, h_i, h_{i+1}, \dots h_{k-1} | h_k)$.

We describe a path on $G_{n,k}$ to $h' = (h_1, \ldots, h_{i+1}, h_i, \ldots, h_k)$, where $1 \leq i < k$.

$$\begin{gathered}
h = (h_1, \ldots, h_i, h_{i+1}, \ldots, h_{k-1}|h_k) \to \\
(h_2, \ldots, h_{i+1}, h_{i+2}, \ldots, h_k|x) \to \\
(h_3, \ldots, h_{i+2}, h_{i+3}, \ldots, x|y) \to \\
\vdots \\
(h_i, h_{i+1}, \ldots, x, y, \ldots, h_{i-4}|h_{i-3}) \to \\
(h_{i+1}, h_{i+2} \ldots, x, y, \ldots, h_{i-3}|h_{i-2}) \to \\
(h_{i+2}, h_{i+3}, \ldots, x, y, \ldots, h_{i-2}|h_{i-1}) \to \\
(h_{i+3}, h_{i+4}, \ldots, x, y, \ldots, h_{i-1}|h_{i+1}) \to \\
(h_{i+4}, h_{i+5} \ldots, x, y, \ldots, h_{i+1}|h_i) \to \\
\vdots \\
(h_1, \ldots, h_{i+1}, h_i, \ldots, h_{k-1}|h_k) = h'
\end{gathered} \tag{4.6}$$

From these adjacent symbol swaps we can construct a transposition that swaps any pair of symbols.

$$\begin{gathered}
(h_1, \ldots, h_i, h_{i+1}, \ldots, h_{j-1}, h_j, \ldots, h_{k-1}|h_k) \to \cdots \to \\
(h_1, \ldots h_{j-1}, h_i, h_j, \ldots h_{k-1}|h_k) \to \cdots \to \\
(h_1, \ldots h_{j-1}, h_j, h_i, \ldots h_{k-1}|h_k) \to \cdots \to \\
(h_1, \ldots, h_j, h_{i+1}, \ldots, h_{j-1}, h_i, \ldots h_{k-1}|h_k)
\end{gathered} \tag{4.7}$$

This completes the proof. □

This bound has an intuitive interpretation in terms of the model. Assuming that $k \geq n-1$, the resulting walks along the history graph is deterministic since there's at most one choice of state at any given stage. When $k \leq n-2$, we can traverse the history graph from any vertex.

4.4 Random Walk on Historised Graph

Let $G_{n,k}$ be a historical graph and let P be a transition probability matrix for $G_{n,k}$. Initially, we consider discrete random walks with respect to P, and extend this to account for any continuous random walk derived from P. Of particular interest in this section is the existence of a stationary distribution for the process governed by P: a distribution π on $V(G_{n,k})$ such that $\pi P = \pi$. We show that for any $n > k+1$, the Markov chain X governed by $G_{n,k}$ converges to π independently of the initial distribution, in the sense that, for any $g \in V(G_{n,k})$, $\lim_{n\to\infty} \mathbb{P}(X_n = g) = \pi_g$. This allows us to make calculations based on the long-term properties of the process without direct simulation and sampling.

The following theorem is a partial result of Perron and Frobenius on the properties real, non-negative, square matrices [20].

Theorem 4.6. *Let M be an irreducible $m \times m$ matrix such that $M_{ij} \geq 0$ for all $1 \leq i, j \leq m$. Then there exists a unique left eigenvector ϕ such that $\phi M = \rho\phi$ and $\phi_i > 0$ for all $1 \leq i \leq m$. Furthermore, $\rho > 0$ and for any eigenvalue λ of M, $|\lambda| \leq \rho$.*

We call ϕ the *Perron vector* of M and, if $\sum_i \phi_i = 1$, we call ϕ the *normalised Perron vector* [21].

We know that $G_{n,k}$ is strongly connected for $k \leq n-2$. For theses values of k, the Markov chain governed by P is irreducible. The following theorem shows that, assuming the additional property of aperiodicity, then the Markov chain will converge to the stationary distribution [22].

Theorem 4.7. *Let X be an aperiodic and irreducible Markov chain on a finite set of states with transition probability matrix P. Then X has a unique positive stationary distribution ϕ such that $\phi P = P$. Furthermore, the limiting distribution π, determined by $\pi_i = \lim_{n\to\infty} \mathbb{P}(X_n = i)$, exists and is equal to the ϕ.*

As a consequence to 4.7 and the, to show that the random walk on $G_{n,k}$ converges to the normalised Perron vector ϕ of P, it is sufficient to show that $G_{n,k}$ is aperiodic.

Theorem 4.8. *For any $n > k-1$, $G_{n,k}$ is aperiodic.*

Proof. For any $k < n-1$, consider a cycle starting from h, reaching some h' with $h'_i = h_i$ for all $i \neq j$ for some $1 \leq j \leq k$, and travelling back to h. This corresponds to the procedure 4.5. Suppose that the length of the path 4.5 is N. Then the cycle from h to h' using procedure 4.5 in both directions is of length $2N$. We introduce a new step in the first half of the cycle as follows.

$$\begin{gathered}
(h_1, \ldots, h_i, \ldots | h_k) \to \\
(h_2, \ldots, h_i, \ldots | x) \to \\
(h_3, \ldots, h_i, \ldots, x | y) \to \\
\vdots \\
(h_i, \ldots, h_k, \ldots | h_{i-2}) \to \\
(h_{i+1}, \ldots, h_k, \ldots | h_{i-1}) \to \\
(h_{i+2}, \ldots, h_k, \ldots | h_j) \to \\
\vdots \\
(h_1, \ldots, h_j, \ldots | h_k)
\end{gathered} \tag{4.8}$$

The cycle that uses 4.8 to reach h' and 4.5 to return to h has length $N+1+N = 2N+1$. Since $\gcd(2N, 2N+1) = 1$, the vertex h is aperiodic, and since h was arbitrary, $G_{n,k}$ is aperiodic. □

We relate this historical graph to the complete graph K_n on $\mathcal{S}$ in the following manner.

Definition 4.9. Let ϕ be a stationary distribution for $G_{n,k}$. Then the stationary distribution for K_n is defined by, for any $v \in K_n$,

$$\phi'_v = \sum_{\substack{g \in G_{n,k} \\ g_k = v}} \phi_g$$

Note that the Markov chain on the historised graph $G_{n,k}$ can also be realised as a Markov chain of order k on the graph K_n. There are many different treatments of these processes, such as those by probability transition tensors [23]. Markov chains with variable length memory have also been studied [24] .The main benefit of the model presented in this thesis is that the relation between states and their memories is clearly articulated by the historised graph's edges. Note also that any given Markov chain of order k is not necessarily well defined on $G_{n,k}$.

Chapter 5

Implementation and Design

5.1 Data Structure

Let X be a non-negative random variable. Then X is an exponential random variable, written $X \sim \exp(\beta)$, if the probability density function of X is

$$f(x;\beta) = \frac{1}{\beta} e^{-\frac{x}{\beta}}$$

Recall the following well-known properties of the exponential distribution.

Lemma 5.1. *For $i = 1, \ldots, n$, let $X_i \sim \exp(\beta_i)$ for some $\beta_i > 0$. Then $\min\{X_i\}_{1 \le i \le n} \sim \exp(\sum_{i=1}^{n} \beta_i)$.*

Lemma 5.2. *For $i = 1, \ldots, n$, let $X_i \sim \exp(\beta_i)$ for some $\lambda_i > 0$. Then for any $1 \le j \le n$,*

$$\mathbb{P}(\min\{X_i\}_{1 \le i \le n} = X_j) = \frac{1}{\beta_j \sum_{i=1}^{n} \frac{1}{\beta_i}}$$

Using these properties, we can construct a dictionary data structure that holds the probability distributions of the time spent at each vertex and the selection of subsequent vertices. Each key is a vertex of the historised graph, and the corresponding value is an array of two lists. The ith item in the first list is another vertex and the weight of the edge connecting this vertex to the key is given by the ith item of the second list. Denote by this dictionary $G_{n,k}^{rep}$.

Let $g \in G_{n,k}$ be the ith visited history of a simulation. Define T_g, the random variable governing the holding time of g_i, as an exponentially distributed random variable with scale parameter $\sum_{h:g \to h} W_{g,h}$. The history vertex to be visited is given by the random variable $M_g = \min\{X_h \sim \exp(W_{g,h}) : g \to h\}$. The law of M_g is determined by 5.2.

```
{
        (1, 2) : ([(2, 3), (2, 4)], [1.61486159787714634, 2.4822239707350986])
        (1, 3) : ([(3, 2), (3, 4)], [1.05561306150611867, 0.7749475346436302])
        (1, 4) : ([(4, 2), (4, 3)], [3.79629391180859257, 2.8437127347575126])
        (2, 1) : ([(1, 3), (1, 4)], [0.07082441225279693, 0.7632134057822547])
        (2, 3) : ([(3, 1), (3, 4)], [1.06575993148071149, 1.8709745951262378])
        (2, 4) : ([(4, 1), (4, 3)], [4.09690976721973108, 1.7172951261131046])
        (3, 1) : ([(1, 2), (1, 4)], [0.30008274552799141, 0.5098170969759579])
        (3, 2) : ([(2, 1), (2, 4)], [1.89179708382379679, 5.4720585343788692])
        (3, 4) : ([(4, 1), (4, 2)], [0.39781665703130753, 2.5146159292958346])
        (4, 1) : ([(1, 2), (1, 3)], [2.22040297962063859, 0.9262582554331954])
        (4, 2) : ([(2, 1), (2, 3)], [3.46703585454729911, 2.1145232860310933])
        (4, 3) : ([(3, 1), (3, 2)], [2.96023643160484128, 1.9602066194906183])
}
```

FIGURE 5.1: An example dictionary $G^{rep}_{4,2}$ representing a 2-avoiding MTD system on 4 states.

5.2 Representation of an Adversary

Inspired by [1], a similarity matrix of the MTD system's states is produced that represents the probability that a pair of states share the same vulnerabilities. These probabilities are used in the construction of adversaries to the system. It is assumed that each adversary can use only one vulnerability and can instantly compromise a state when presented. Given a set of states $\mathcal{S}$, let $\mathcal{S}^s$ be the similarity matrix of the states.

To construct a set of adversaries, samples can be taken from a Bernoulli distribution using $\mathcal{S}^s$ as the parameter. Assume that $i \in \mathcal{S}$ has a vulnerability v_i. Then a state $j \neq i$ has vulnerability v_i if the sample from $X \sim$Bernoulli$(\mathcal{S}^s_{i,j})$ is 1. This simulates the uncertainty of the existence of day-0 vulnerabilities: assuming that each state contains a vulnerability, the similarity will dictate whether this vulnerability is shared by other states. Each adversary

Denote by $\mathcal{A}(\mathcal{S}^s)$ a set of tuples $a \subsetneq \mathcal{S}$ for some vulnerabilities. These represent some of the possible adversaries to the MTD system.

5.3 Simulation

Given a simulation time t^{sim}, a dictionary representation $G^{rep}_{n,k}$ of a MTD graph $G_{n,k}$ and a fixed set of adversaries $\mathcal{A}(\mathcal{S}^s)$, the simulation is started by the random uniform selection of an initial vertex $g \in V(G_{n,k})$.

During the mth iteration of the simulation loop, the next vertex v^m is chosen along with the lifetime of the new vertex t_{v^m} in the manner described in section 5.1. Recorded in

a tuple are the current state of the vertex and the time, (v_k^m, t_{v^m}). The loop continues as long as $\sum_m t_{v^m} < t^{sim}$. Let $m^{sim} = \min\{m : \sum_{l=1}^m t_{v^l} \geq t^{sim}\}$.

For each adversary $a = (a_1, \ldots, a_m) \in \mathcal{A}(\mathcal{S}^s)$ associate a *total compromise time*

$$t_a = \sum_{m:v^m \in a} t_{v^m}.$$

and define the *maximum consecutive compromise time*

$$t_a^{con} = \sup_{i \leq j}\{\sum_{m=i}^{j} t_{v^m} : v^m \in a \quad \forall i \leq m \leq j\}$$

5.3.1 Combined Fitness Function

Let $\alpha \in [0, 1]$ then

$$\alpha m^{sim} + (1 - \alpha) \sum_{a \in \mathcal{A}(\mathcal{S}^s)} t_a \tag{5.1}$$

is a candidate combined fitness value for a simulation, but note that the value of each term is unbounded. A normalising function such as 5.2 can be used to restrict the range of m^{sim} to a suitable set of values while preserving the monotonicity of m^{sim}.

$$1 - \frac{1}{m^{sim}} \in [0, 1] \tag{5.2}$$

Likewise, the function measuring time compromised can be restricted to this same range by substituting t_a for t_a/t^{sim} and taking an average over the number of adversaries.

$$\frac{1}{t^{sim}|\mathcal{A}(\mathcal{S}^s)|} \sum_{a \in \mathcal{A}(\mathcal{S}^s)} t_a \in [0, 1] \tag{5.3}$$

This results in a modified equation 5.4 that takes values in $[0, 1]$.

$$\alpha\Big(1 - \frac{1}{m^{sim}}\Big) + \frac{1 - \alpha}{t^{sim}|\mathcal{A}(\mathcal{S}^s)|} \sum_{a \in \mathcal{A}(\mathcal{S}^s)} t_a \tag{5.4}$$

Equation 5.4 possesses on obvious benefit over equation 5.1: the restriction of 5.4 to a compact range allows an appropriate choice of weight α to be made without needing to consider possible explosion effects in the fitness values obtained.

An equation similar to 5.4 can be made by substituting either function for t_a^{con}. Furthermore, for any partition (α, β, γ) of 1, a combined fitness function can be created, where the partition serves as weights of the individual functions.

The biggest drawback to the weighted single-objective optimisation problem is that, by projecting the information obtained by the fitness function $(f_1, \ldots, f_m)$ onto $\mathbb{R}$, information is lost and it is possible for the intermediate solutions produced by the algorithm to be constrained to local minima [25]. Instead, the results obtained in this thesis are produced via multi-objective optimisation. This heuristic produces its own problems, predominantly that the execution time will be slower . To help with this, approximations of each component in the fitness function are developed to circumvent simulation and aggregation of data.

5.4 Deterministic Approximation of Simulation

The most obvious drawback of direct simulation of this kind is that a representative sample of time compromised and number of transitions requires a simulation time that is at least proportional to $n(n-1)\cdots(n-k+1) = |E(G_{n,k})|$. The time cost of simulations for larger values of n and k necessitates a more direct calculation that considers the global system as a whole in place of a series local properties.

5.4.1 Approximating m^{sim}

Approximating the number of transitions a given n, k-MTD system will make can be done by using the properties of both the stationary distribution and the weights of each history. Since the random walk on $G_{n,k}$ converges to the Perron vector ϕ, given any history g, the frequency with which g is visited tends to ϕ_g. The rate at which the system transitions away from g is given by the random variable $T_g \sim \exp\left(\sum_{h:g\to h} W_{g,h}\right)$. By 5.1, $\mathbb{E}[T_g] = (\sum_{h:g\to h} W_{g,h})$. We therefore have that

$$\mathbb{E}[m^{sim}](t^{sim}) = \frac{t^{sim}}{\sum_{g\in G_{n,k}} \phi_g \mathbb{E}[T_g]} \tag{5.5}$$

Since 5.5 is monotonically increasing with respect to t^{sim}, for the purposes of comparison between possible solutions it is enough to set $t^{sim} = 1$.

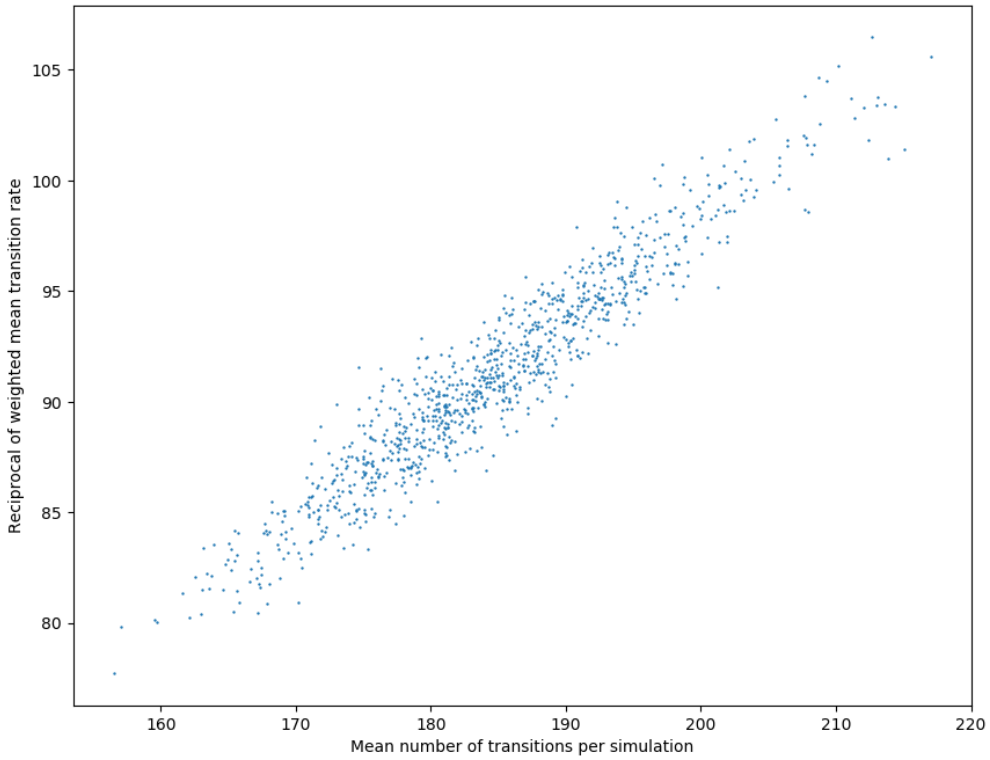


FIGURE 5.2: Plot of the mean number of transitions of 10 simulations run for 500 units of time versus the approximation in terms of the stationary distribution of 1000 random $6,3$-MTD graphs, where $|\mathcal{A}| = 10$ is randomly generated. The Pearson correlation of this set of data is 0.955.

5.4.2 Approximating t_a

To construct an approximation for t_a we take inspiration from Chung's paper on the Cheeger Inequality for Directed Graphs [26]. Let W be the matrix of weights of the edges for the graph $G_{n,k}$ and let P be the corresponding probability transition matrix of the form

$$P_{g,h} = \frac{W_{g,h}}{\sum_{z:g\to z} W_{g,z}} = \frac{W_{g,h}}{\mathbb{E}[T_g]} \quad , \quad \forall u, v \in V(G_{n,k}).$$

Recall 4.6, which implies that for each $G_{n,k}$ with $n > k-1$ the normalised Perron vector ϕ exists and is unique.

Definition 5.3. A circulation on a digraph G is a function $F : E(G) \to \mathbb{R}_{\geq 0}$ such that

$$\sum_{u:u\to v} F(u,v) = \sum_{w:v\to w} F(v,w)$$

It is easy to check that $F_\phi(u,v) = \phi_u P_{u,v}$ is indeed a circulation [26].

Definition 5.4. Let $s \in V(G_{n,k})$, $\emptyset \neq S \subsetneq V(G_{n,k})$ and $\partial S = \{u \in S : \exists v \in S^c, u \to v\}$.

1. $F_\phi(s) = \sum_{u:u\to s} F_\phi(u,s)$,
2. $F_\phi(S) = \sum_{v\in S} F_\phi(v)$,
3. $F_\phi(\partial S) = \sum_{u\in S, v\notin S} F_\phi(u,v)$.

Definition 5.5. For any $a \in \mathcal{A}(\mathcal{S}^s)$, let $S_a = \{g \in V(G_{n,k}) : g_k \in a\}$. Define the *Perron compromised circulation* by

$$F_\phi(\mathcal{A}(\mathcal{S}^s)) := \frac{1}{|\mathcal{A}(\mathcal{S}^s)|} \sum_{a\in\mathcal{A}(\mathcal{S}^s)} \frac{F_\phi(\partial S_a)}{F_\phi(S_a^c)}. \tag{5.6}$$

where $S_a^c = \{v \in V(G_{n,k}) : v \notin S_a\}$ is the complement of S_a.

Equation 5.6 uses the ratio between the boundary circulation and the outer circulation of the non-compromised vertices. The more vertices that are compromised, the smaller the value of the denominator hence the larger $F_\phi(\mathcal{A}(\mathcal{S}^s))$ is and vice versa. Likewise, if the boundary from non-compromised to compromised vertices has a large circulation then the system will become compromised more frequently. Note that $F_\phi(\partial S) = F_\phi(\partial S^c)$.

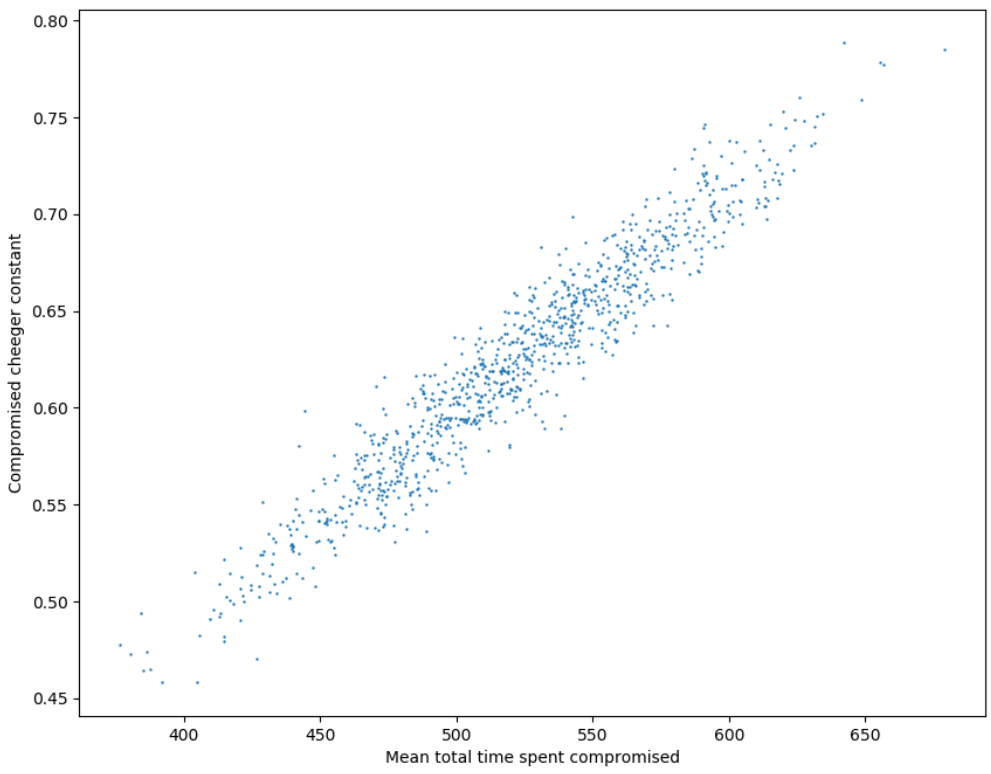


FIGURE 5.3: Plot of the mean time spent compromised of 10 simulations run for 500 units of time versus the Perron compromised circulation of 1000 random $6,3$-MTD graphs, where $|\mathcal{A}| = 10$ is randomly generated. The Pearson correlation is 0.961.

Notice also that $F_\phi(\mathcal{A}(\mathcal{S}^s))$ is bounded above by 1 and achieves this bound whenever $\mathcal{A}(\mathcal{S}^s) = \{a : |a| = n-1\}$. A simulation of 1000 randomly generated 4,2-MTD graphs shows that 5.6 and $\mathbb{E}[t^{sim}]$ are highly correlated (see figure 5.2).

5.4.3 Approximating t_a^{con}

The approximation of t_a^{con} is accomplished via an algorithm run on the set of vertices and a given set of bugs. For each $a \in \mathcal{A}(\mathcal{S}^s)$, vertices in $V(G_n, k)$ are labelled as compromised if their current state is in a. For each vertex $v \in V(G_{n,k})$, all walks from v on compromised vertices that end with an uncompromised vertex x are considered. Let $(v, v_1, v_2, \ldots, v_n, x)$ be such a path. The probability of following this path is given by

$$\phi_v \mathbb{P}(X_1 = v_1 | X_0 = v)\mathbb{P}(X_{n+1} = x | X_n = v_n) \prod_{i=1}^{n-1} \mathbb{P}(X_{i+1} = v_{i+1} | X_i = v_i), \tag{5.7}$$

where X is the discrete time Markov chain, and the expected time taken is given by

$$\mathbb{E}\left[T_v + \sum_{i=1}^{n-1} T_{v_i} + T_x\right] \tag{5.8}$$

The statistic is obtained by multiplying 5.7 by 5.8 . To prevent infinite loops, a sensitivity ϵ is chosen such that the path's value is ignored if the statistic is smaller than ϵ. An average is taken over all possible paths from all vertices and over all $a \in \mathcal{A}(\mathcal{S}^s)$ to obtain a real number. The correlation of this number to simulated values can be seen in figure 5.4. Algorithm 1 displays pseudocode for this procedure.

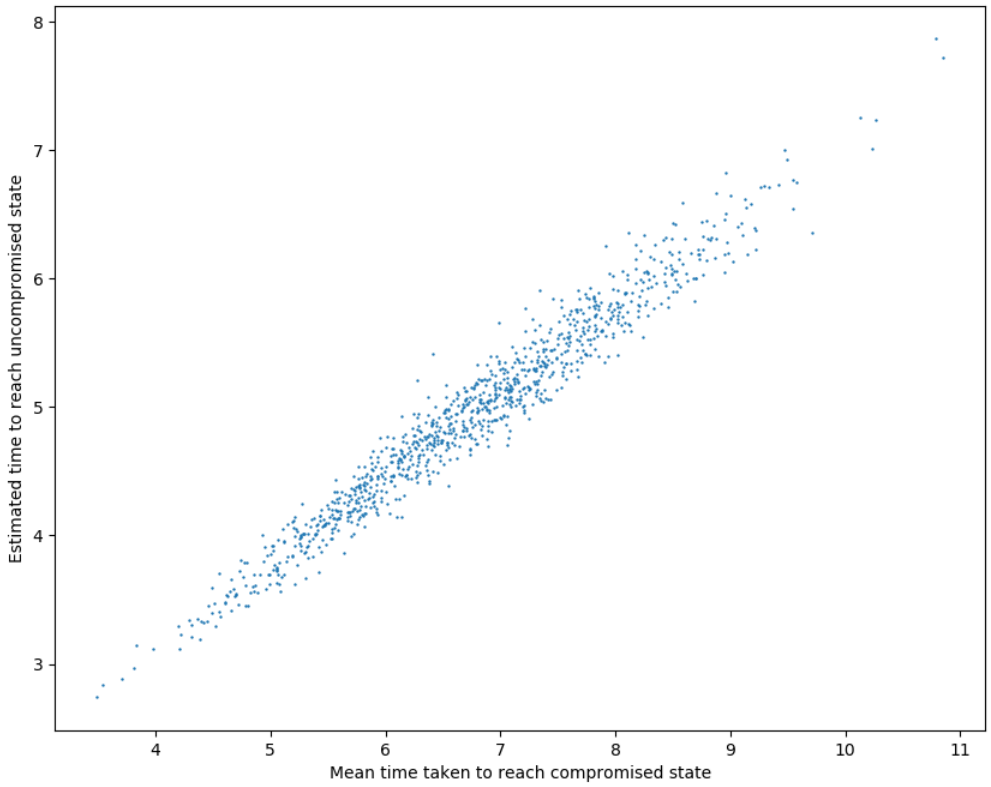


FIGURE 5.4: Plot of the mean time compromised per bug of 10 simulations run for 500 units of time versus the approximation using equations 5.7 and 5.8 for 1000 $6,3$-MTD graphs with sensitivity 0.001, where $|\mathcal{A}| = 10$ is randomly generated. The Pearson correlation of this set of data is 0.977.

5.5 Pareto Optimisation

The biggest drawback to the single value fitness functions presented in the previous subsection is that information about the system is lost when multi-dimensional values are mapped to $\mathbb{R}$. A possible solution to this is multi-objective, or Pareto, optimisation.

Definition 5.6. Consider the multi-objective optimisation problem

$$\min_{x \in X} \big(f_1(x), f_2(x), \ldots, f_m(x)\big), \tag{5.9}$$

where X is the set of possible solutions to the problem, usually defined by a set of constraints, and $f_i : X \to \mathbb{R}$. A feasible solution $x^1 \in X$ is said to *Pareto-dominate* a solution $x^2 \in_{\text{Par}} X$, writing $x^1 \prec x^2$ if

1. $f_i(x^1) \leq f_i(x^2)$ for all $1 \leq i \leq m$,
2. $f_j(x^1) < f_j(x^1)$ for some $1 \leq j \leq m$.

Algorithm 1: Approximation of time until non-compromise.

Input: A weighted digraph $G = (V, E, W)$ with probability transition matrix P and stationary distribution ϕ, a set of bugs $\mathcal{A}$ and a sensitivity ρ.

Output: A real number correlated with the time taken for the system to reach an uncompromised state.

```
begin
    t ← 0
    foreach a ∈ 𝒜 do
        t_a ← 0;
        foreach u ∈ V do
            t_u ← 0;
            if u ∈ a then
                continue;
            end
            M ← ∅;
            foreach v ∈ V do
                if (u, v) ∈ E then
                    M ← M ∪ {v, P_uv, 𝔼[T_u]};
                end
            end
            while M ≠ ∅ do
                m ← M[0];
                M ← M \ {m};
                if m[0] ∉ a then
                    t_u ← t_u + m[1] * m[2];
                    if m[1] * m[2] > ρ then
                        foreach w ∈ V do
                            if (v, w) ∈ E then
                                M ← M ∪ {w, m[1] * P_v,w, m[2] + 𝔼[T_u]};
                            end
                        end
                    end
                end
            end
            t_a ← t_a + t_u * φ_u;
        end
        t ← t + t_a;
    end
    return t/|𝒜|;
end
```

A solution x is said to be *Pareto-efficient* if there exists no solution x' that Pareto dominates x. The set of Pareto efficient solutions is known as the *Pareto front*, denoted $Par(X)$.

Let $k < n - 1$. The domain of possible solutions $X_{n,k}$ considered in the optimisation problem 5.9 will be the rectangular subset of $\mathbb{R}^{|\mathcal{E}_{n,k}|}$ bounded by the lines $x_i = 1$, $x_i = 10$ for all $1 \leq i \leq |\mathcal{E}_{n,k}|$. The lower bound is set to prevent blow-up in equation 5.5, and the upper bound is enforced to prevent arbitrarily large values from being created that could prevent the algorithm from halting. The fitness function $f = (f_1, f_2, f_3)$ is used, where f_1, f_2, f_3 are the approximations obtained in subsections 5.4.1, 5.4.2, 5.4.3 respectively.

5.6 Multi-objective Genetic Algorithm

In order to search through the domain for suitable solutions, a genetic algorithm search heuristic is employed. The algorithm used is a near-faithful implementation of FPGA (Fast Pareto Genetic Algorithm), which is shown to outperform the widely popular NSGA-II (Nondominated Sorting Genetic Algorithm) when the fitness values are computationally expensive [27, 28].

5.6.1 Outline of Algorithm

To initialise the algorithm, a starting population $\mathbf{P}_0$ is uniformly randomly created. A member $p_{0,i}$ of $\mathbf{P}_0$ is created by sampling uniformly from the space of solutions X. Furthermore, a maximum population size M, a perturbation value ρ, a generational threshold N and a population sensitivity ϵ. All data obtained in this thesis uses the values $M = 500$, $\rho = 20$, $N = 15$, $\epsilon = 0.05$.

Each member of the population $p_{0,i}$ has their fitness value $f(p_{0,i})$ calculated. The function used here is an array of the approximations from the previous section. These values are then used as a basis for ranking $\prec_{\text{Front}}$ the members of $\mathbf{P}_0$ which is a combination of algorithm 2 combined with a further sort of the members of the first front using algorithm 3 which replaces the size of dominating set with distance between points for maximal variation [28].

The main loop of the algorithm begins. At step m, deterministic binary tournament is then used to select the population used to create the next generation. Two members of the population $p_{m,i}, p_{m,j}$ are selected chosen uniformly and if $p_{m,i} \prec_{Front} p_{m,j}$, then $p_{m,i}$ is chosen and vice versa. In the event that they are equally ranked, one is chosen uniformly randomly. The winner of each tournament is appended to the crossover set C_m until its cardinality reaches the population size.

Crossover is conducted by randomly selecting two members of C_m and executing algorithm 4, creating a set of offspring O_m with cardinality 20. The fitness ranking of the members of O_m are calculated and a combined population $CP_m = P_m \cup O_m$ is formed and ranked according to $\prec_{Front}$.

Before committing to another loop, the stopping criteria are updated. This algorithm calculates the value $\frac{|C_m|}{|D_m|}$, where $D_m = \{p \in CP_m : \nexists q(q \prec_{FRONT} p)\}$ is the set of non-dominated solutions in CP_m. The purpose of this value is to approximate the capacity of the algorithm to produce new non-dominated solutions: consecutive values close to

Algorithm 2: Fast non-dominated sort [28].

Input: Set of graphs P with fitness function pre-calculated.
Output: Partition of results into several fronts.

```
begin
    F_1 ← ∅;
    foreach p ∈ P do
        _p ← ∅;
        s_p ← 0;
        foreach q ∈ P do
            if p ≺_Par q then
                D_p ← D_p ∪ {q}
            else
                if q ≺_Par p then
                    s_p ← s_p + 1;
                    F_1 ← F_1 ∪ {p};
                end
            end
        end
        if s_p = 0 then
            F_1 ← F_1 ∪ {(p, |D_p|, 0)}
        end
    end
    i ← 1;
    while F_i ≠ ∅ do
        Q ← ∅;
        foreach p ∈ F_i do
            foreach q ∈ D_p do
                s_q ← s_q − 1;
                if s_q = 0 then
                    Q ← Q ∪ {(q, |D_q|, s_q)};
                end
            end
        end
        i ← i + 1;
        F_i ←Q;
    end
    return F_1, F_2, . . . ;
end
```

1 indicate the algorithm is unable to create new non-dominated solutions. Once N consecutive generations have values within ϵ distance of 1, the algorithm terminates.

If the stopping criteria are not met, the population size of the next generation is calculated using the formula

$$|P_{m+1}| = \min\{20 + |D_m|, M\} \tag{5.10}$$

Without prior knowledge of the exact description of the Pareto front, convergence of the algorithm to this set is difficult to prove. The authors of this algorithm have conducted

Algorithm 3: Sorting algorithm of solutions in Pareto front [28].

Input: Subet of graphs P in the Pareto front with fitness function array F that evaluates $p \in P$.
Output: Values for each $p \in P$ that can be used to order P.

```
begin
    d ← ∅;
    l ← |P|;
    foreach p ∈ P do
        d_p ← 0;
        d ← d ∪ {d_p}
    end
    foreach f ∈ F do
        P ← sort(P, f);
        P[0] ← ∞;
        P[l − 1] ← ∞;
        foreach i ∈ {1, . . . , l − 2} do
            d_P[i] ← d_P[i] + (f(P[i + 1]) − f(P[i − 1])) / |f(P[0]) − f(P[l − 1])|;
        end
    end
    return d;
end
```

Algorithm 4: Crossover algorithm analogous to Simulated Binary Crossover [29], using the modification for bounded domains [30]

Input: Two members p, q of C_m with weights W_p, W_q, pertubation value ρ and lower and upper bounds X_L, X_U.
Output: Two new solutions p', q'.

```
begin
    l ← |W_p|;
    W_1 ← ∅;
    W_2 ← ∅;
    foreach i ∈ {0, . . . , l − 1} do
        u ← rand(0, 1);
        β ← 1 + 2 / {|W_1[i] − W_2[i]|} min{ min{W_1[i], W_2[i]} − x_L, x_U − max{W_1[i], W_2[i]} };
        α ← 2 − β^(ρ+1);
        if u ≤ 1/α then
            γ ← (αu)^(1/(ρ+1));
        else
            γ ← (1/(2 − αu))^(1/(ρ+1));
        end
        W_1[i] ← 1/2 (W_1 + W_2 − γ|W_2 − W_1|);
        W_2[i] ← 1/2 (W_1 + W_2 + γ|W_2 − W_1|);
    end
    return W_1, W_2;
end
```

extensive testing, benchmarking the algorithm against NSGA-II and testing on functions with known Pareto fronts [27].

5.7 Comparison to Random Samples

The first test of this algorithm is conducted by calculating the Pareto fronts for small n and comparing these to the fitness values of digraphs with randomly-generated weights. See figure 5.5 for a combined plot of these values.

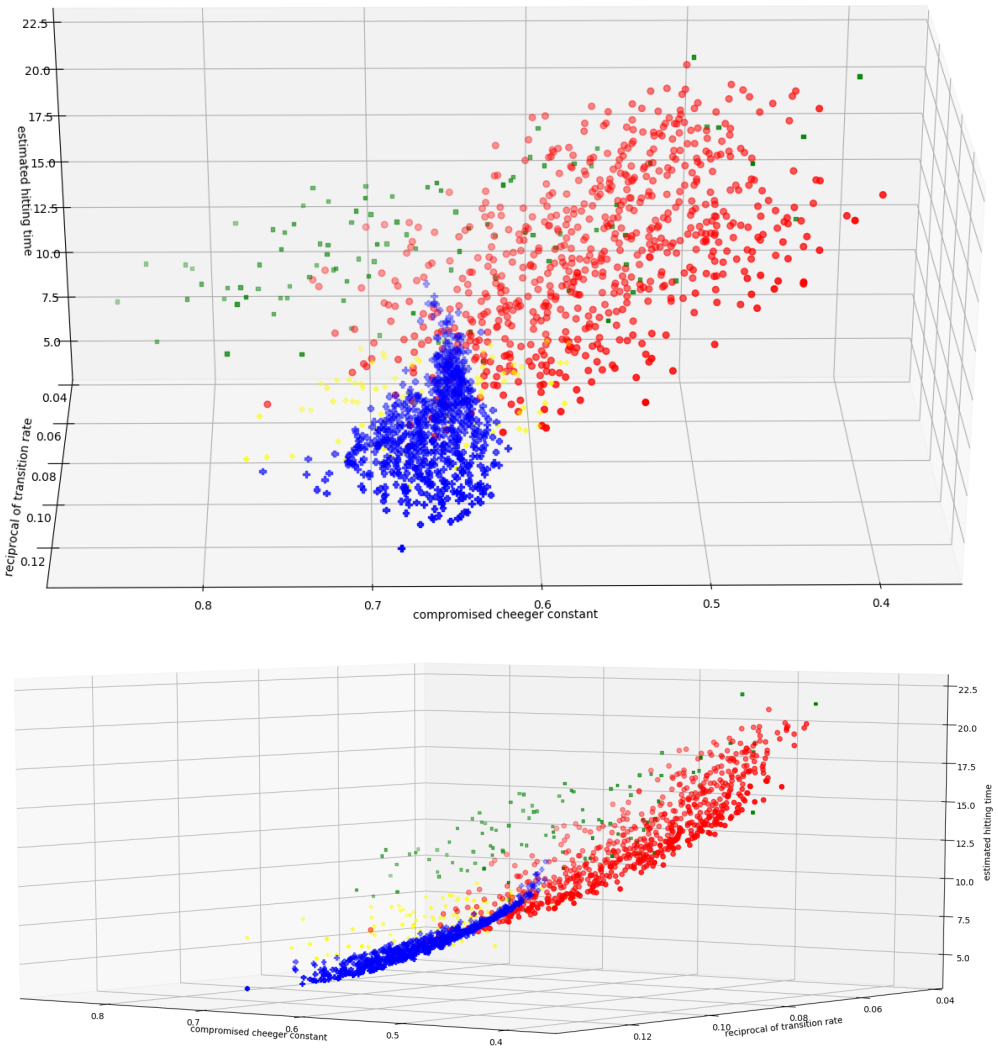


FIGURE 5.5: Combined plot of Pareto fronts for $n = 4, k \in \{1, 2\}$, represented by red circles and blue crosses respectively, and 100 random $4, 1$- and $4, 2$-graphs, represented by green squares and yellow diamonds. Observe that most, if not all, green squares and yellow diamonds are dominated by the red circles and blue crosses. The single adversary used compromised the states (1,2).

Chapter 6

Data Analysis and Results

Observing the 3-dimensional plots in figure 6.1, some questions, with associated statistical hypotheses, can be formulated. Let $\mathrm{Par}(X_{n,k})$ be the Pareto front associated to n and k.

1. In each dimension, do the mean and variances of the Pareto front vary as k varies?
2. Is there a relationship between f_2 and f_3 that varies independently of f_1? That is, does there exist a function $g : \mathbb{R} \to \mathbb{R}$ such that $g\big(f_2(x)\big) \approx f_3(x)$?
3. Does the standard deviation of f_1 decrease with k?

6.1 Comparison of Pareto Fronts for different values of k

To explore the relationship between Pareto fronts, an arbitrary value $n = 6$ is chosen with adversary set $\mathcal{A} = \{(1,2,3),(1,3,4),(2,5),(4,5),(2,3,5)\}$. The Pareto fronts for $k \in \{1,2,3,4\}$ are calculated. Treated as a sample from a probability distribution, the statistics of these fronts are summarised in tables 6.1, 6.2, 6.3, where f_i refers to the value of the ith component of the fitness value of members of the Pareto front.

Prior to formulating the hypothesis test comparing the mean value of each front, a suitable distribution for each front is found. For this, Cullen and Frey graphs are calculated using the `fitdistrplus` package in R, which can be found in appendix A. It is found that the distribution of f_1 is relatively normal, but f_2 and f_3 cannot be considered so. As such, a non-parametric Kruskal-Wallis test is conducted on all sets of values, with a further ANOVA test on f_1. For both tests, the null hypothesis used is that the mean of all samples are equal. All Kruskal-Wallis tests returned p-values at of at most 7×10^{-32},

and the one-way ANOVA test returned a p-value of 1.9×10^{-18}, confirming that all fronts have different means for all f_1, f_2, f_3.

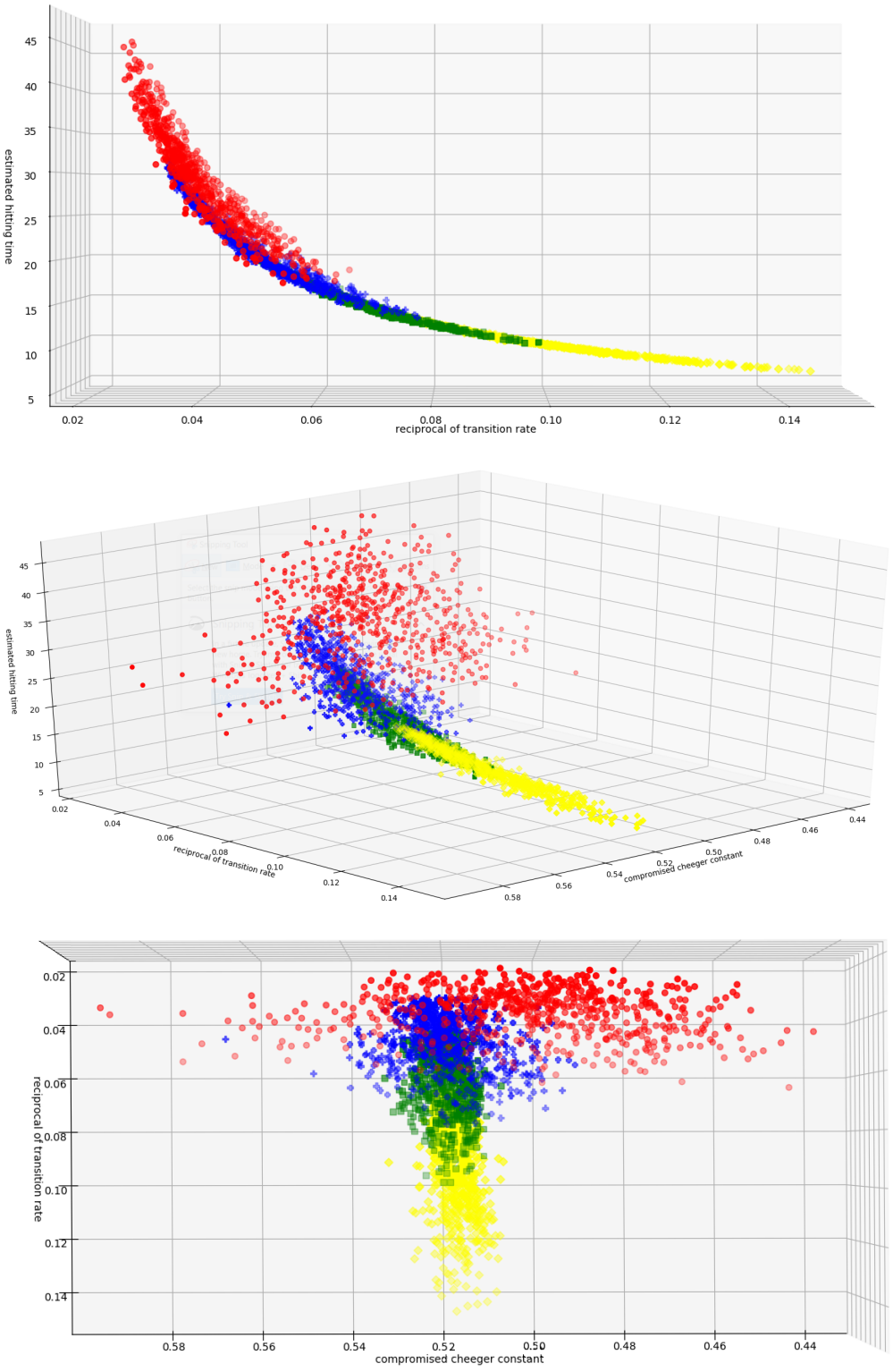


FIGURE 6.1: Combined plot of Pareto fronts for $n = 6, k \in \{1, 2, 3, 4\}$, represented by red circles, blue crosses, green squares and yellow diamonds respectively.

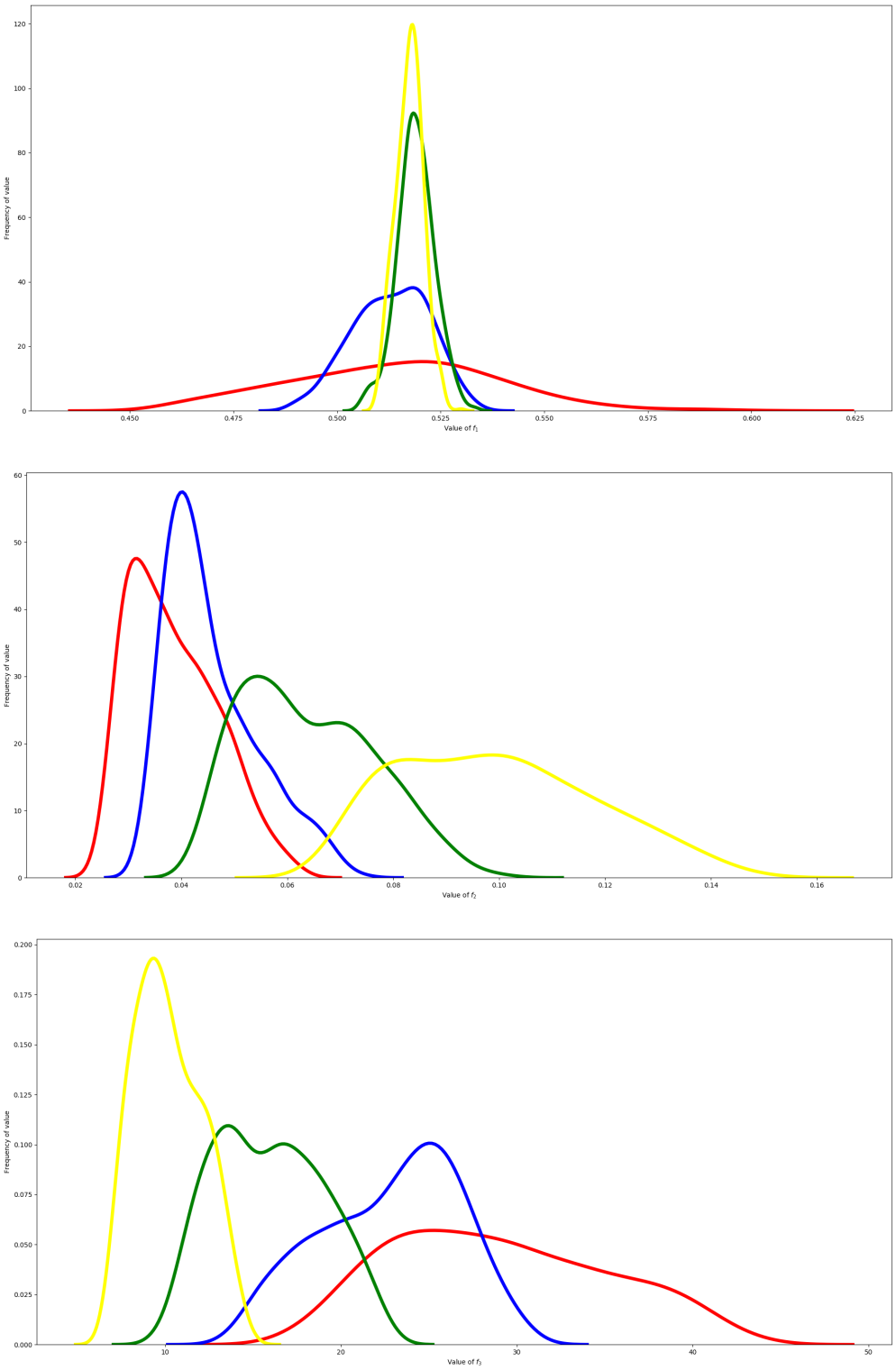


FIGURE 6.2: Combined plots of value density of fitness function components f_1, f_2, f_3 for $n = 6, k \in \{1, 2, 3, 4\}$, represented by red, blue, green and yellow curves respectively.

TABLE 6.1: Statistical properties of first fitness component of members of the Pareto front by value of k

k	#	f_1					
		mean	std.dev.	min	max	skew	kurtosis
1	701	0.514	0.0255	0.457	0.457	0.122	-0.0782
2	701	0.513	0.00922	0.489	0.489	-0.205	-0.548
3	700	0.519	0.00453	0.505	0.505	-0.0767	0.350
4	698	0.517	0.00332	0.509	0.509	0.0411	-0.130

TABLE 6.2: Statistical properties of second fitness component of members of the Pareto front by value of k

k	#	f_2					
		mean	std.dev.	min	max	skew	kurtosis
1	701	0.0386	0.00832	0.0253	0.0253	0.559	-0.508
2	701	0.0457	0.00862	0.0331	0.0331	0.924	0.111
3	700	0.0640	0.0124	0.0439	0.0439	0.419	-0.695
4	698	0.0995	0.0186	0.0663	0.0663	0.355	-0.686

TABLE 6.3: Statistical properties of third fitness component of members of the Pareto front by value of k

k	#	f_3					
		mean	std.dev.	min	max	skew	kurtosis
1	701	29.1	6.17	16.6	16.6	0.278	-0.842
2	701	22.8	3.90	13.5	13.5	-0.299	-0.815
3	700	15.9	3.07	9.68	9.68	0.169	-0.954
4	698	10.2	1.88	6.52	6.52	0.284	-0.834

6.2 Correlation of f_2 and f_3

Upon observing the first 3-dimensional plot of estimations seen in figure 6.1, it is easy to notice a strong correlation between the reciprocal of the transition rate f_2 and the estimated hitting time of a non-compromised vertex f_3. A Spearman test outputs a correlation coefficient of -0.9959398750287742, confirming this relation. A curve-fitting library `scipy.optimize.curve_fit` was used on the function $g(x) = \frac{a}{x} + b$ for constants a, b to be determined. The values $a = 1.098482, b = -1.46434885$ were identified, which produce the fit seen in figure 6.3.

$$f_3(x) = \frac{1.098482}{f_2(x)} - 1.46434885 \tag{6.1}$$

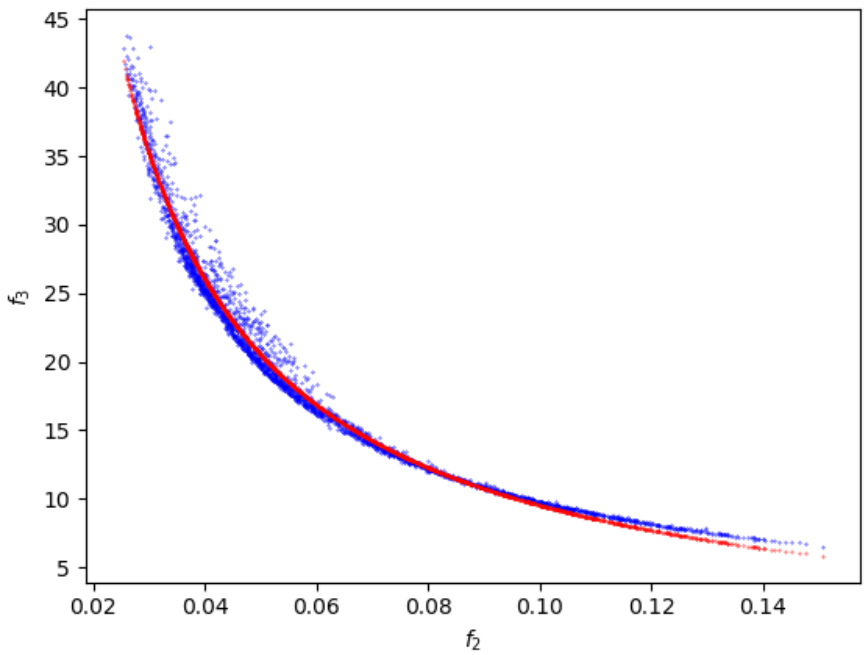


FIGURE 6.3: Plot of (f_2, f_3) components of all Pareto fronts (blue) against the approximation with equation 6.1 (red).

6.3 Relationship between Standard Deviation of f_1 and k

Since the distribution of f_1 is approximately normal (see A), Barlett's test for equality of more than two variances is used, which tests against the null hypothesis of equality of variances between all samples. The test statistic returned is 3260, corresponding to a vanishingly small p-value, indicating non-equality of variances.

6.4 Discussion

In this section, it was shown that each front has a different mean for all of it's fitness functions, therefore each value of k gives rise to different properties with respect to security and efficiency. The general trend is that lower values of k correspond to optimal solutions with less frequent transitions but with more time spent being potentially compromised by each adversary. Furthermore, lower values of k correspond to a higher spread of total time spend being potentially compromised.

Note also that, since the cardinality of each Pareto front is large (approximately 700), the statistical power of all hypothesis tests considered is strong [31], which explains the small p-values obtained for each test.

Chapter 7

Conclusion and Further Work

7.1 Conclusion

In this thesis, a model for Moving Target Defence systems was developed that generalised previous models to the more general continuous-time setting, which allows for the analysis of efficiency along with security. The model was a Markov chain of fixed order k on n states, referred to as a weighted digraph. A set of objective functions, f_1, f_2, f_3, were created that are approximations of the simulation statistics of proportion of time spend potentially compromised, number of transitions and time until the system reaches a non-compromised state per adversary. A genetic algorithm was developed that, given a set of adversaries, would output an optimal set of weighted digraphs.

The structure of this set was then statistically analysed. It was shown the fronts varied with k, corresponding to a change in properties with k: the degree to which the Markov chain remembers it's history, assuming membership to the corresponding Pareto front, increases the number of transitions and the security of the system as k increases. Furthermore, the mean time spent compromised has a lower variance at higher values of k.

The statistical analysis conducted characterises the trade-off between security and efficiency inherent to the model. This characterisation allows for the existence of very efficient dynamic defences, but also illustrates the lack of security afforded by such a system.

Since the choice of adversary set $\mathcal{A}$ is arbitrary, it is possible to recalculate the Pareto fronts to account for a change in adversaries. A real-world example of the need for such a mechanic is for the discovery of a new vulnerability that doesn't contribute to the similarity of the states.

7.2 Further Work

This research is a first step in the development of efficient Moving Target Defence systems.

7.2.1 Theoretical Development of Front Geometry

The analysis conducted in this thesis was restricted to the case $n = 6$, with a fixed set of adversaries. While the general shape of each front and the relationship between them has been discussed for this case, the more general case of arbitrary n has not been explored. Further research can be conducted into showing, possibly proving, that the statistical properties shown in this thesis hold in the more general case.

7.2.2 Development of Sophisticated Adversary Models

The model for adversaries presented in this thesis is perhaps the most simple model possible: each adversary can exploit the system using one bug that can be applied to different states as determined by their similarity. Furthermore, no distinction is made between a states potential to be compromised and the event that a successful hack is achieved. More complex models would enhance any of these properties.

The concept of a similarity matrix was used by earlier models to predict the probability of an adversary successfully compromising a system [1]. More exotic models might consist of multiple similarity matrices specific to each bug. This could be used to simulate more specific vulnerabilities: two systems might both be vulnerable to DDoS attacks, whereas only one of the two is vulnerable to SQL injection.

7.2.3 Varying values of k

For any set of states $\mathcal{S}$, a digraph $G_{n,k}$ can be created with an associated continuous-time Markov chain $Y = Y_{t,k}$. Suppose that k is allowed to vary with t. A step function $k(t)$ can be created that takes values in $\{1, \ldots, n-2\}$, such that at time t, the stochastic process $Y_{t,k(t)}$ is determined by $Y_{t,k}$, where $k = k(t)$. This would create a Markov chain of varying order, who's properties have been studied previously [24].

The first problem to be resolved is to define, for $1 < k, l < n-1$, how a transition occurs between the digraphs $G_{n,k}$ and $G_{n,l}$. Take $k > l$ and suppose the system transitions at time τ - $k(t) = k$ for $t < \tau$ and $k(t) = l$ for $t \geq \tau$. Then for $t < \tau$, $Y_{n,k(t)} = (g_1, g_2, \ldots g_{k-1}|g_k)$. One might define, for $t \geq \tau$, $Y_{n,k(t)} = (g_{k-l+1}, \ldots, g_{k-1}|g_k)$. The case

where $k < l$ requires an additional rule of how a k-history is mapped to a single l-history. Initially, one might attempt to keep the full memory of the previous states and take the l most recent states to form a vertex of $G_{n,l}$. This procedure does not always produce valid vertices: as an explicit example, the sequence $(1, 2, 3) \to (2, 3, 4) \to (3, 4, 1)$ is a path in $G_{7,3}$, but this procedure will produce $(1, 2, 3, 4, 1)$, which is not a valid vertex of $G_{7,5}$.

To analyse the long-time properties of such a stochastic process, the convergence in distribution of $Y_{t,k(t)}$ to a unique stationary distribution needs to be verified. Furthermore, the approximations of each of the statistics will need to be reformulated to accommodate the differing sets of vertices and weights.

The aim of such a system would be to take advantage of the properties that each process $Y_{t,k}$ possesses, some of which were discussed in chapter 6. A realistic scenario for this could be for systems that are able to consistently detect compromises as they happen - some mechanisms have been proposed [32–34]. For example, a system might occupy a high-efficiency state, corresponding to a smaller value of $k(t)$, until a higher security configuration is required, corresponding to a relative increase in $k(t)$.

7.2.4 Layered Moving Target Defence

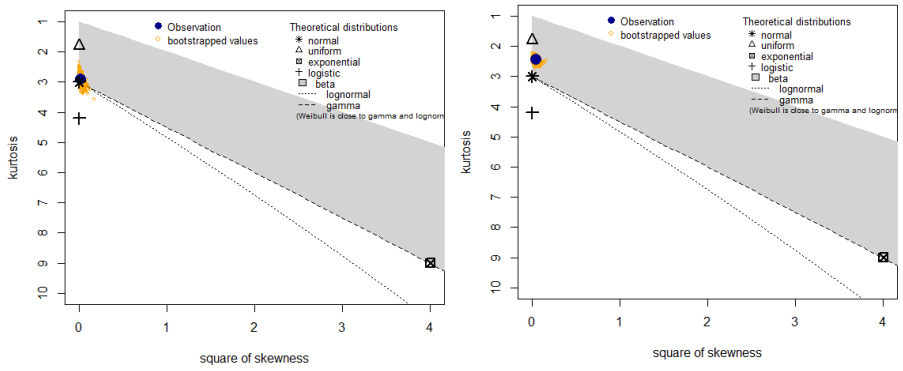
An earlier study of the game played between a defending MTD system and an adversary explored the use of multiple layers as additional protection against compromise, as discussed in subsection 2.1.2 [2].

7.2.5 Nash Equilibria

Studies have attempted to find and classify the Nash equilibria of certain MTD game scenarios [3, 35, 36]. In the interest of consistency, the behaviour of Nash equilibria when the system is controlled by an automaton (the Markov chain) should be explored. The equilibria of games with players who's strategies are determined by random determinist finite automata has been a topic of interest since at least 1986, when Ariel Rubinstein described their behaviour playing the classic Prisoner's Dilemma game [37]. Any such study will certainly be dependant on the complexity of the adversary.

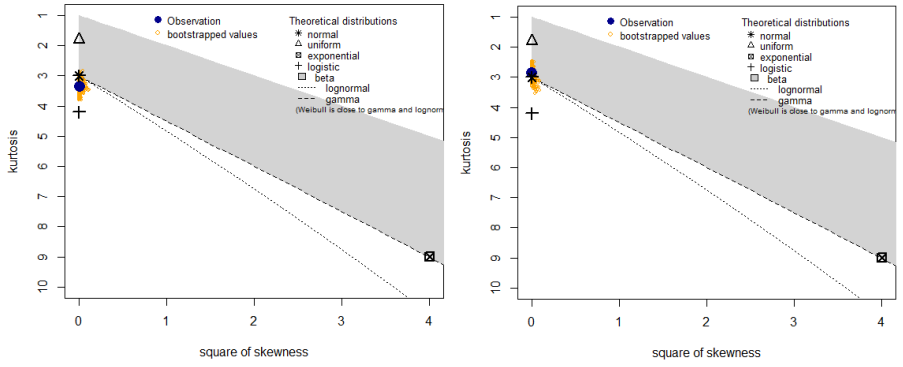
Appendix A

Cullen and Frey graphs



(a) k=1

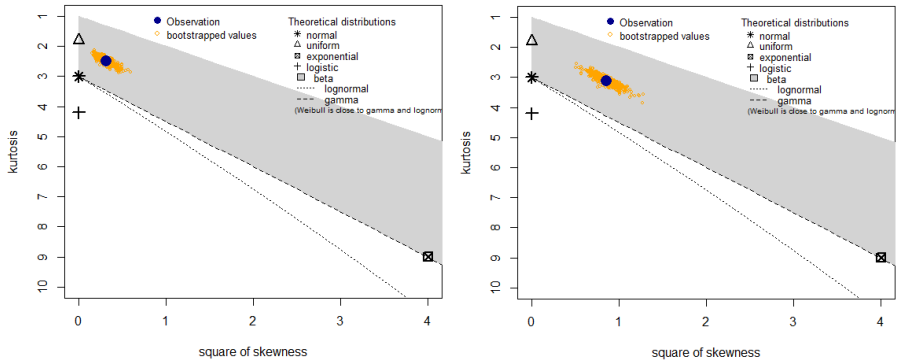
(b) k=2



(c) k=3

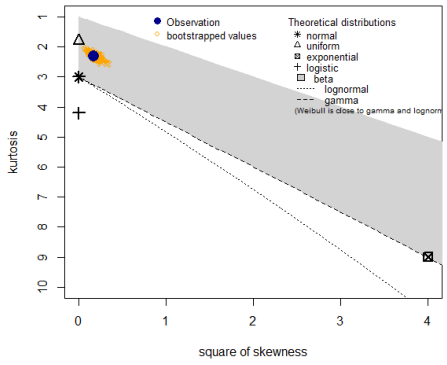
(d) k=4

FIGURE A.1: Cullen and Frey plots of distribution of f_1 values for $n = 6, k \in \{1, 2, 3, 4\}$ with 500 accompanying bootstraps.

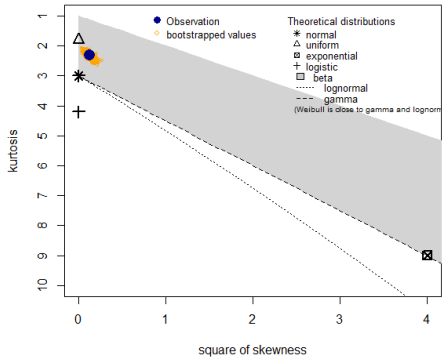


(a) k=1

(b) k=2

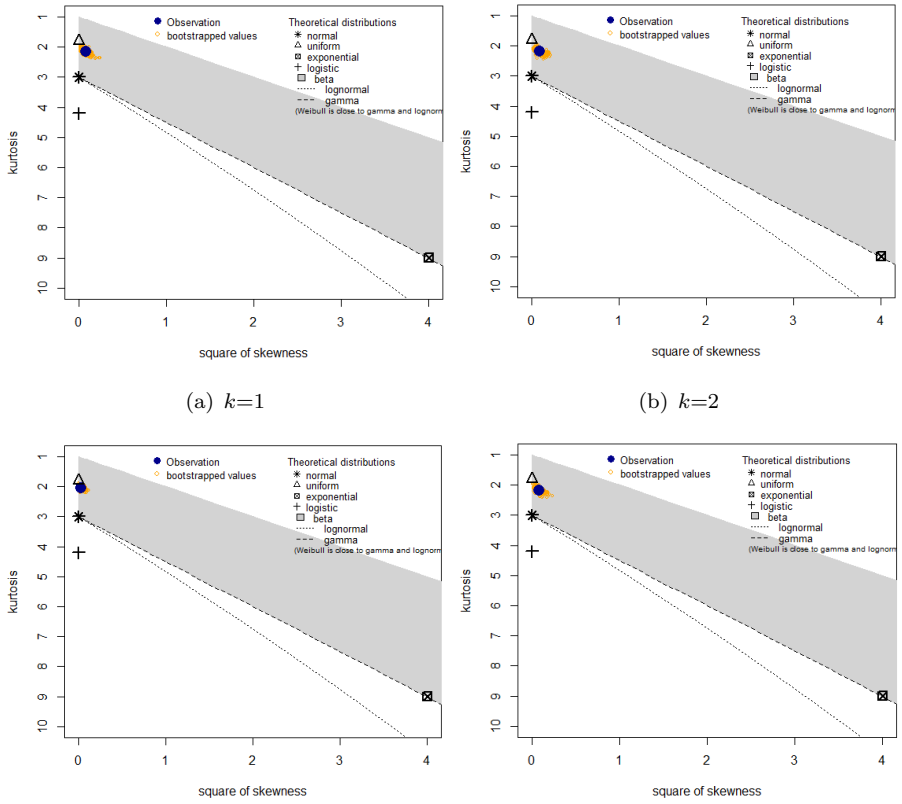


(c) k=3



(d) k=4

FIGURE A.2: Cullen and Frey plots of distribution of f_2 values for $n = 6, k \in \{1, 2, 3, 4\}$ with 500 accompanying bootstraps.



(a) k=1 (b) k=2

(c) k=3 (d) k=4

FIGURE A.3: Cullen and Frey plots of distribution of f_3 values for $n = 6, k \in \{1, 2, 3, 4\}$ with 500 accompanying bootstraps.

Bibliography

[1] Kevin M. Carter, James F. Riordan, and Hamed Okhravi. A game theoretic approach to strategy determination for dynamic platform defenses. In *Proceedings of the First ACM Workshop on Moving Target Defense*, MTD '14, pages 21–30, 2014. ISBN 978-1-4503-3150-0. doi: 10.1145/2663474.2663478.

[2] Hoda Maleki, Saeed Valizadeh, William Koch, Azer Bestavros, and Marten van Dijk. Markov modeling of moving target defense games. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, MTD '16, pages 81–92, 2016. ISBN 978-1-4503-4570-5. doi: 10.1145/2995272.2995273.

[3] Achintya Prakash and Michael P. Wellman. Empirical game-theoretic analysis for moving target defense. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, MTD '15, pages 57–65, 2015. ISBN 978-1-4503-3823-3. doi: 10.1145/2808475.2808483.

[4] Louis Marinos, Adrian Belmonte, and Evangelos Rekleitis. ENISA threat landscape 2015. Technical report, ENISA, Jan 2016. URL https://www.enisa.europa.eu/publications/etl2015/. Accessed: 27/05/2019.

[5] Fraser Howard. A closer look at the angler exploit kit. Technical report, Sophos Group plc, July 2015. URL https://news.sophos.com/en-us/2015/07/21/a-closer-look-at-the-angler-exploit-kit/. Accessed: 27/05/2019.

[6] Nick Biasini. Threat spotlight: Cisco Talos thwarts access to massive international exploit kit generating $60m annually from ransomware alone. Technical report, Cisco Talos Intelligence Group, Oct 2015. URL https://www.talosintelligence.com/angler-exposed. Accessed: 27/05/2019.

[7] Cisco Systems Inc. Cisco 2015 midyear security report. Technical report, 2015. URL https://www.cisco.com/c/dam/assets/global/UK/events/switchup_challenge/pdf/cisco-msr-2015.pdf. Accessed: 27/05/2019.

[8] Michael Hopkins and Ali Dehghantanha. Exploit kits: The production line of the cybercrime economy? *CoRR*, abs/1707.04714, 2017. URL http://doi.org/10.1109/InfoSec.2015.7435501. Accessed: 27/05/2019.

[9] Sushil Jajodia and Kun Sun. MTD 2014: First ACM workshop on moving target defense. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1550–1551, 2014. ISBN 978-1-4503-2957-6. doi: 10.1145/2660267.2660385.

[10] NITRD. National cyber leap year summit 2009 co-chairs' report. Technical report, National Office for the Federal Networking and Information Technology Research and Development Program, Sept 2009. URL https://www.nitrd.gov/nitrdgroups/images/b/bd/National_Cyber_Leap_Year_Summit_2009_CoChairs_Report.pdf. Accessed: 270/05/2019.

[11] B.C. Ward, S.R. Gomez, R.W. Skowyra, D. Bigelow, J.N. Martin, J.W. Landry, and H. Okhravi. Survey of cyber moving targets. Technical report, Massachusetts Institute of Technology, Lexington, Massachusetts, Jan 2018.

[12] Mason Wright, Sridhar Venkatesan, Massimiliano Albanese, and Michael P. Wellman. Moving target defense against ddos attacks: An empirical game-theoretic analysis. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, MTD '16, pages 93–104, 2016. ISBN 978-1-4503-4570-5. doi: 10.1145/2995272.2995279.

[13] Kimberley D. Seed. Battling phages: How bacteria defend against viral attack. *PLOS Pathogens*, 11(6):1–5, 06 2015. doi: 10.1371/journal.ppat.1004847. URL https://doi.org/10.1371/journal.ppat.1004847.

[14] David J. John, Robert W. Smith, William H. Turkett, Daniel A. Cañas, and Errin 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, pages 1261–1268, 2014. ISBN 978-1-4503-2881-4. doi: 10.1145/2598394.2605437.

[15] Adetokunbo Makanju, A. Nur Zincir-Heywood, and Shinsaku Kiyomoto. On evolutionary computation for moving target defense in software defined networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 287–288, 2017. ISBN 978-1-4503-4939-0. doi: 10.1145/3067695.3075604.

[16] Robert Walter Smith. Evolutionary strategies for secure moving target configuration discovery. Master's thesis, Wake Forest University Graduate School of Arts and Sciences, Winston-Salem, North Carolina, 2014.

[17] S. Wang, L. Zhang, and C. Tang. A new dynamic address solution for moving target defense. In *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, pages 1149–1152, May 2016. doi: 10.1109/ITNEC.2016.7560545.

[18] Gordon Slade. Self-avoiding walk on the complete graph. 2019.

[19] G Grimmett. Probability on graphs: Random processes on graphs and lattices. 2010.

[20] Oskar Perron. Zur theorie der matrices. *Mathematische Annalen*, 64:248–263, 1907. URL http://eudml.org/doc/158317.

[21] Roger A. Horn and Charles R. Johnson, editors. *Matrix Analysis*. Cambridge University Press, USA, 1985. ISBN 0521305861.

[22] Richard Serfozo. *Basics of Applied Stochastic Processes, Springer 2009*. 01 2009. doi: 10.1007/978-3-540-89332-5.

[23] Sheng-Jhih Wu and Moody T. Chu. Markov chains with memory, tensor formulation, and the dynamics of power iteration. *Applied Mathematics and Computation*, 303:226 – 239, 2017. ISSN 0096-3003. doi: https://doi.org/10.1016/j.amc.2017.01.030. URL http://www.sciencedirect.com/science/article/pii/S0096300317300383.

[24] Martin Mächler and Peter Bühlmann. Variable length markov chains: Methodology, computing, and software. *Journal of Computational and Graphical Statistics*, 13(2): 435–455, 2004. doi: 10.1198/1061860043524. URL https://doi.org/10.1198/1061860043524.

[25] H. Ishibuchi, Y. Nojima, and Tsutomu Doi. Comparison between single-objective and multi-objective genetic algorithms: Performance comparison and performance measures. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1143–1150, July 2006. doi: 10.1109/CEC.2006.1688438.

[26] Fan Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9(1):1–19, Apr 2005. doi: 10.1007/s00026-005-0237-z. URL https://doi.org/10.1007/s00026-005-0237-z.

[27] Hamidreza Eskandari and Christopher Geiger. A fast pareto genetic algorithm approach for solving expensive multiobjective optimization problems. *J. Heuristics*, 14:203–241, 06 2008. doi: 10.1007/s10732-007-9037-z.

[28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197, April 2002. ISSN 1941-0026. doi: 10.1109/4235.996017.

[29] Ram Agrawal, Kalyanmoy Deb, and Ram Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9, 06 2000.

[30] Kalyanmoy Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2):311 – 338, 2000. ISSN 0045-7825. doi: https://doi.org/10.1016/S0045-7825(99)00389-8. URL http://www.sciencedirect.com/science/article/pii/S0045782599003898.

[31] Norman R. Draper. The cambridge dictionary of statistics, fourth edition by b. s. everitt, a. skrondal. *International Statistical Review*, 79(2):273–274, 2011. doi: 10.1111/j.1751-5823.2011.00149_2.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1751-5823.2011.00149_2.x.

[32] Christopher Oehmen, Elena Peterson, and Scott Dowson. An organic model for detecting cyber-events. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, CSIIRW '10, pages 66:1–66:4, 2010. ISBN 978-1-4503-0017-9. doi: 10.1145/1852666.1852740.

[33] Christopher Oehmen, Elena Peterson, and Jeremy Teuton. Evolutionary drift models for moving target defense. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, CSIIRW '13, pages 37:1–37:4, 2013. ISBN 978-1-4503-1687-3. doi: 10.1145/2459976.2460018.

[34] Ankur Chowdhary, Sailik Sengupta, Dijiang Huang, and Subbarao Kambhampati. Markov game modeling of moving target defense for strategic detection of threats in cloud networks. *CoRR*, abs/1812.09660, 2018. URL http://arxiv.org/abs/1812.09660.

[35] Abdel Rahman Eldosouky, Walid Saad, and Dusit Niyato. Single controller stochastic games for optimized moving target defense. *2016 IEEE International Conference on Communications (ICC)*, May 2016. doi: 10.1109/icc.2016.7510985. URL http://dx.doi.org/10.1109/ICC.2016.7510985.

[36] C. Lei, D. Ma, and H. Zhang. Optimal strategy selection for moving target defense based on markov game. *IEEE Access*, 5:156–169, 2017. ISSN 2169-3536. doi: 10.1109/ACCESS.2016.2633983.

[37] Ariel Rubinstein. Finite automata play the repeated prisoner's dilemma. *Journal of Economic Theory*, 39(1):83 – 96, 1986. ISSN 0022-0531. doi: https://doi.org/

10.1016/0022-0531(86)90021-9. URL http://www.sciencedirect.com/science/article/pii/0022053186900219.