Evolving Security Policies

Yow Tzu Lim

Doctor of Philosophy

THE UNIVERSITY of York

Department of Computer Science

2010

To my parents, Kong Woon and Ai Lim

Abstract

As computer system size and complexity grow, formulating effective policies require more sophistication. There are many risk factors that need to be considered, some of which may be in conflict. Inevitably, unpredictable circumstances that demand decisions will arise during operation. In some cases an automated response may be imperative; in other cases these may be ill-advised. Manual decisions are often made that override the current policy and serve effectively to redefine it. This matter is further complicated in highly dynamic operational environments like mobile ad-hoc networks, in which the risk factors may be changing continually. Thus, security policies must be able to change and adapt to the operational needs.

This study investigates the potential of evolutionary algorithms as a tool in determining the optimal security policies that suit such environments. This thesis reviews some fundamental concepts in related domains. It presents three applications of evolutionary algorithms in solving problems that are of direct relevance. These include the inference of security policies from decision examples, the dynamic adaptation of security policies, and the optimisation of security policies for a specific set of missions. The results show that the inference approaches based on evolutionary algorithms are very promising.

The thesis concludes with an evaluation of the work done, the extent to which the work justifies the thesis hypothesis and some possible directions on how evolutionary algorithms can be applied to address a wider range of relevant problems in the domain of concern.

Contents

1 Introduction **1**

1.1 Motivation . . . 2

1.1.1 Many risk factors to be considered . . . 2

1.1.2 Operational needs change . . . 2

1.1.3 Deriving effective security policies is hard . . . 3

1.1.4 One size (policy) does not fit all . . . 4

1.2 Technical approach . . . 5

1.3 Thesis hypothesis . . . 8

1.4 Thesis organisation . . . 9

2 Computer Security **11**

2.1 Computer security objectives . . . 12

2.1.1 Confidentiality . . . 12

2.1.2 Integrity . . . 13

2.1.3 Availability . . . 13

2.1.4 Authenticity . . . 14

2.1.5 Accountability and non-repudiation . . . 15

2.1.6 Summary . . . 15

2.2 Security risk analysis . . . 16

2.2.1 Assets, vulnerabilities and threats identification . . . 16

2.2.2 Risk assessment . . . 17

2.2.3 Selection of controls . . . 18

2.2.4 Re-evaluation . . . 19

2.2.5 Summary . . . 19

2.3 Dynamic coalitions and MANETs 20
2.3.1 MANETs 21
2.3.2 Security challenges of MANETs 23
2.4 Conclusions 25

3 Security Policy Models 26
3.1 Mandatory access control policy models 27
3.1.1 Bell-LaPadula model 27
3.1.2 Biba model 29
3.1.3 Chinese wall model 29
3.1.4 Clark-Wilson model 31
3.2 Discretionary access control policy models 31
3.3 Role based access control policy models 32
3.4 Flexible access control policy models 34
3.4.1 Fuzzy MLS model 35
3.4.2 Economics based models 40
3.4.3 Top-down hierarchical models 48
3.5 Summary 54
3.6 Conclusions 55

4 Learning Techniques 56
4.1 Evolutionary algorithms (EAs) 56
4.1.1 Evaluation 57
4.1.2 Selection 57
4.1.3 Reproduction 59
4.1.4 Implementations of EAs 59
4.2 Genetic Programming (GP) 60
4.2.1 Extensions on GP 61
4.3 Grammatical Evolution (GE) 63
4.3.1 Genotype-phenotype mapping 64
4.3.2 Extensions on GE 66
4.4 Multi-objective evolutionary algorithms (MOEAs) 67
4.4.1 Weighted sum of fitness functions 67

4.4.2 Pareto front based approaches 69
4.4.3 Advantages of Pareto front based approaches 71
4.4.4 Implementations of Pareto front based approaches 71
4.5 Relevant applications of GP and GE 74
4.5.1 Rule inference system 74
4.5.2 Intrusion and anomaly detection systems 75
4.5.3 Security protocols 75
4.5.4 Cryptography 76
4.6 Fuzzy expert systems 76
4.6.1 Fuzzy set theory 76
4.6.2 Inference process 79
4.7 Conclusions 80

5 Static Policy Inference 82
5.1 Experimentation on binary decision policies 83
5.1.1 Experiment 5.1: Partial MLS Bell-LaPadula policy 87
5.1.2 Experiment 5.2: Full MLS Bell-LaPadula policy 92
5.1.3 Experiment 5.3: Budgetised MLS policy 94
5.2 Experimentation on multi-decision policies 95
5.2.1 Experiment 5.4: Rule based approach 98
5.2.2 Experiment 5.5: Regression based approach 100
5.2.3 Experiment 5.6: Grammatical evolution 101
5.2.4 Experiment 5.7: Fuzzy set ensemble 102
5.2.5 Experimental results and evaluation 106
5.3 Example Security Policies Inferred 111
5.4 Evidence for the thesis and future work 112
5.5 Conclusions 115

6 Dynamic Policy Inference 116
6.1 Data stream classification 117
6.2 A dynamic security policy model 119
6.3 Experimentation 122
6.3.1 Static policy learning 123

6.3.2 Dynamic policy learning 131
6.4 Evidence for the thesis and future work 144
6.5 Conclusions 149

7 Mission-specific Policy Discovery 150
7.1 Scenario: travelling across a battlefield 151
7.1.1 Movement strategy 152
7.1.2 Risk-budget based security policy 157
7.2 Experimentation 159
7.2.1 Experiment 7.1: Minimising casualty toll with f_{dead} 160
7.2.2 Experiment 7.2: Minimising casualty toll with f_{msd} 160
7.2.3 Experiment 7.3: Minimising casualty toll with DOO 161
7.2.4 Experiment 7.4: Multi-objective optimisation 161
7.3 Experimental results and evaluation 162
7.4 Example security policies inferred 167
7.5 Evidence for the thesis and future work 167
7.6 Conclusions 172

8 Evaluation and conclusions 173
8.1 Evaluation 173
8.1.1 Static policy inference 174
8.1.2 Dynamic policy inference 175
8.1.3 Mission-specific policy discovery 176
8.1.4 Thesis contributions 176
8.2 Envisaged future work 177
8.2.1 Policy fusion 177
8.2.2 The robustness of a security policy 178
8.2.3 Scalability with the training set size 179
8.2.4 More complex security policies 179
8.2.5 More complex scenarios 180
8.2.6 Other deployment domains 180
8.3 Closing remarks 180

References 181

List of Tables

3.1 The differences between the model based on the command economic system and the model based on the market economic system. 45

5.1 The experimental result summary of the inferred policies using GP and GE. 110

5.2 Some of the optimal security policies inferred with GP and GE. . 111

6.1 The values of α and its changes, $\delta\alpha$, at every 500 generations. . . 135

7.1 The experimental result summary on the performances of the optimal policies with respect to the casualty toll of blue agents found using GP/MOGP and DOO. 164

List of Figures

2.1 The conceptual model of a security risk analysis process [1]. . . . 20

3.1 An access control matrix example. 32
3.2 The Fuzzy MLS model [2]. 36
3.3 The ITA policy model [3]. 51

4.1 A GP individual that represents a formula $(X \times Y) + (4 - (Y + 1))$. 60
4.2 The crossover and mutation operations in GP. 62
4.3 A mapping between solution space S to objective space O. 68
4.4 The set of solutions that lie at the concave part of the feasible region can never be discovered. 69
4.5 The objective of a Pareto front based approach is to approximate the Pareto front of the solutions with the individuals in the population. 70
4.6 The archive truncation process in SPEA2 removes the non-dominated individuals that are closest to others *iteratively* until these individuals can fit into the archive. Assuming the archive size is 10, the truncation process removes the individuals that are crossed out. . 73
4.7 Various interpretations of the fuzzy operations. 78
4.8 The inference process of a fuzzy expert system [4]. 81

5.1 Some well typed individuals that represent the conditions in the MLS Bell-LaPadula policy model. 85
5.2 Some examples of inferred policies in Experiments 5.1. 89

5.3 The predefined values for each M_j based on the triangular fuzzy membership function. 104
5.4 The predefined values for each M_j based on the Gaussian distribution curve. 105
5.5 The output risk bands of the target reference model defined in (5.10). 113
5.6 The output risk bands of the inferred policies shown in Table 5.2. 114

6.1 The distributions of the best individuals in all three experiments at different error rate intervals. 128
6.2 Some examples of the security policies learnt. 129
6.2 Some examples of the security policies learnt. 130
6.3 The median error rate of the best policies learnt in Experiment 6.4. 134
6.4 In the DOO setting, all individuals are at the Pareto front. SPEA2 removes individuals that are closest to others *iteratively* until they can fit into the archive. The individual at each corner is guaranteed to be in the archive. 138
6.5 The median error rates of the best policies learnt in Experiment 6.4 (light pink) and Experiment 6.5 (dark blue). 139
6.6 The median error rates of the best policies learnt in Experiment 6.5 (light pink) and Experiment 6.6 (dark blue). 142
6.7 The median error rates of the ultimate ensemble models constructed from the models learnt in Experiment 6.4. 145
6.8 The median error rates of the ultimate ensemble models constructed from the models learnt in Experiment 6.5. 146
6.9 The median error rates of the ultimate ensemble models constructed from the models learnt in Experiment 6.6. 147

7.1 The risk posed to each square by a red agent located at the centre. 154
7.2 The mapping between the indices of a *DSM* and the movement directions they represent. 154
7.3 The simulated battlefield grid map. 155
7.4 The non-dominated solutions in one run using different initial budgets. 165
7.5 The Pareto optimal solution set. 166

7.6 The optimal policy with respect to the casualty toll of blue agents. 168
7.7 The optimal policy with respect to the mission completion time. . 169
7.8 The optimal policy with respect to the casualty toll of blue agents given the constraint that the mean of mission completion time has to be within 70 time steps. 170

Acknowledgements

I would like to express my gratitude to all who have provided support in accomplishing this thesis. I would like to thank my Ph.D. supervisor, John A. Clark, for the opportunity given to carry out this research and also for his guidance, expertise and encouragement during the course of this research. I would like to thank John A. McDermid and Robert I. Damper for their critical comments and suggestions given in their roles as the assessors.

I am grateful for having had two internship opportunities and one research visit to the IBM T. J. Watson Research Center in Hawthorne, New York, and be acquainted with many bright and nice people. They have provided insightful feedback and suggestions regarding my work. In particular, I would like to thank my manager Pankaj Rohatgi and my mentor Pau Chen Cheng. They have not only guided me with their expertise, but also trained me to think critically through one-to-one discussions and helped to build my confidence in having my own ideas and in presenting them to others. Many thanks also go to Wei Fan and Charu C. Aggarwal, who have provided invaluable insight and direction in the data stream mining domain and to Vugranam C. Sreedhar, Charanjit S. Jutla and Suresh N. Chari in the security domain.

I would like to thank the research members in my department, especially Juan E. Tapiador, Sevil Sen, David R. White and Chen Hao, who have provided constructive discussions on my research. I would also like to thank the support staff, especially Mark Hewitt, Aaron Turner and James Carter, who have maintained the grid servers, on which my experiments have been carried out. A special thank you also goes to Grace Shiao En Chng in spending her valuable time to proofread this thesis.

I would also like to express my appreciation to my wife Ka Ki Lai, my parents Kong Woon Lim and Ai Lim Chum, my sisters Sien Sin Lim, Wan Sin Lim and Kai Xin Lim, for their endless support and care in my personal life during the course of my Ph.D. study. I would like to thank God Almighty, who has given me the strength and peace to work hard on this project.

Lastly, I am indebted to the US Army Research laboratory and the UK Ministry of Defence for the provision of financial support under the Project 6: Trust and Risk Management in Dynamic Coalition Environments in the International Technology Alliance in Network and Information Sciences (ITA) Project.

Declaration

The work submitted in this thesis is the result of my own investigation. Work appearing here has appeared in print as follows:

- Yow Tzu Lim, Pau Chen Cheng, John A. Clark, and Pankaj Rohatgi. IBM Research Report RC24442: Policy Evolution with Genetic Programming. Technical report, 2007.
- Yow Tzu Lim, Pau Chen Cheng, John A. Clark, and Pankaj Rohatgi. Policy Evolution with Genetic Programming: A Comparison of Three Approaches. In *2008 IEEE World Congress on Computational Intelligence*, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.
- Yow Tzu Lim, Pau Chen Cheng, Pankaj Rohatgi, and John A. Clark. MLS Security Policy Evolution with Genetic Programming. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1571–1578, Atlanta, GA, USA, 12-16 July 2008. ACM.
- Yow Tzu Lim, Pau Chen Cheng, John A. Clark, and Pankaj Rohatgi. Policy Evolution with Grammatical Evolution. In *The Second Annual Conference of the International Technology Alliance*, Adelphi, Maryland, USA, 2008.
- Yow Tzu Lim, Pau Chen Cheng, John A. Clark, and Pankaj Rohatgi. Policy Evolution with Grammatical Evolution. In *Proceedings of the 7th International Conference on Simulated Evolution And Learning (SEAL '08)*, volume 5361 of *Lecture Notes in Computer Science*, pages 71–80, Melbourne, Australia, December 7-10 2008. Springer.

- Yow Tzu Lim, John A. Clark, Pau Chen Cheng, and Juan E. Tapiador. Mission Specific Security Policy Discovery. In *The Third Annual Conference of the International Technology Alliance*, Adelphi, Maryland, USA, 2009.
- Yow Tzu Lim, Pau Chen Cheng, John A. Clark, and Pankaj Rohatgi. IBM Research Report RC24865: Dynamic Security Policy Learning. Technical report, 2009.

I am the primary author of all work reported in this thesis. Advice on specific aspects of the work was provided by my Ph.D. supervisor, John A. Clark and other co-authors.

The research was sponsored by US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorised to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

Chapter 1

Introduction

Information security is not a new concept. Information has always been of value and has been something to protect. The computer age has merely increased the volume and nature of information stored and processed, and has provided enhanced opportunities for creating and accessing that information. As computational equipment becomes increasingly embedded in the fabric of our environment and information processing permeates ever more aspects of our lives, we are forced to continually assess the risk taken with respect to such information.

We have now moved significantly away from the mainframe computer era. Much modern computation is built around highly distributed resources. There are many advantages to providing resources in a distributed fashion. However, securing resources in such environments is distinctly non-trivial.

A recent development in this area has been the emergence of dynamic coalitions. Whereas present distributed systems might comprise a variety of agents and nodes that we know about and largely have control over, dynamic coalitions may arise where parties with little or no experience of each other need to work together to achieve their goals. These coalitions give rise to significant issues concerning how the risk of interaction may feasibly and effectively be managed.

A significant tool in the risk management of existing system has been *security policies*. Security policies are often defined to restrict the information accesses to restricted groups of users. For example, the Multi-Level Security (MLS) Bell-LaPadula model [5] is concerned with the information confidentiality in a computer system processing classified information. This model is a very simply stated and implementable policy. It has been adopted in military computer systems.

1.1 Motivation

However, as system size and complexity grow, creating and implementing effective policies require more sophistication. In the context of large distributed systems, the concept of policy hierarchy has been introduced in [6]. This concept suggests that high-level policies can be derived from business goals and refined into low-level policies, which can then be executed by the system [6, 7]. Since then, there has been a proliferation of research work carried out in this domain, mainly focussing on the security policy analysis and refinement processes. This has led to the birth of many policy refinement models, languages and tools.

1.1.1 Many risk factors to be considered

Even with the aid of these tools, formulating an optimal security policy remains a difficult problem because there are many factors that need to be considered, some of which are in conflict. The tradeoffs among these factors are often made by the security administrators based on their experiences and intuitions. Inevitably, there is some degree of subjectivity and arbitrariness in this assessment. The security risk analysis may also be incomplete and some risk factors may be left out of consideration.

1.1.2 Operational needs change

Additionally, in current practice, security is typically managed in terms of fixed, rigid security policies. There is a growing acceptance that current security mechanisms are not appropriate for many future network systems, for example, mobile ad-hoc networks (MANETs). This problem has recently received increasing attention in the research community. Inevitably, unpredictable circumstances that demand decisions will arise during operation. In some cases an automated response may be imperative; in other cases these may be ill-advised. Manual decisions are often made that override the current policy and serve effectively to redefine it. (It is accepted that a policy may not be suited to all circumstances, and in particular, not suited to the current one.) This matter is further complicated in highly dynamic operational environments like MANETs, where the risk factors are constantly changing. A new requirement is thus needed: the security policy has to be able to change and adapt to the operational needs or else will inevitably be circumvented in creative ways (often referred to as workarounds) [2, 8].

A common solution to this problem is manually creating exceptions to policies by granting the required permissions to the users to meet operational needs [8]. This process can be tedious and time consuming. Worse, the exceptions granted are often never revoked when they should be [8]. People with goals to meet will find ways around policies. For example, information is sometimes classified at a lower sensitivity level than it should be to facilitate information sharing. The security policy is thus being tweaked and used in such a way as to fit the operational needs. A more principled and conceptually clear approach taking this requirement into account would therefore be advantageous.

1.1.3 Deriving effective security policies is hard

The interaction between the security administrators and the decision makers is often able to elicit, at best, a partial description of the security policy. These stakeholders may well be able to give specific decisions to specific instances of authorisation requests, but may not always be adept at generalising these decisions to a security policy, especially when the complexities of MANETs are taken into account. The matter is further complicated by the fact that the operational benefit as well as risk must eventually be taken into account. There is currently no coherent way forward on this issue. It would be very useful to be able to codify what a “principled basis” consists of, since this serves to document “good practice” and facilitates its propagation. We might ask whether we can leverage the knowledge, experience and decision-making abilities of the stakeholders in a creative fashion to elicit more generally applicable security policies. We shall address this issue later.

1.1.4 One size (policy) does not fit all

In addition to these problems, the security requirements may differ on a case-by-case basis. Recent research [8] has provided excellent articulation of why precanned one-size-fits-all security policies and mechanisms are inappropriate to many modern day operations. Many more abstract frameworks for access control systems are being proposed to cope with the realities of modern systems. An example of these is the risk-budget based approach [8]. In this framework, users are given risk budgets that they can spend to access information. Riskier accesses cost more. We believe such approaches raise several issues: what initial budgets should be given? Should we allow an unfettered free market? If not, then what constraints should we impose? Should there be minimum costs associated with specific accesses?

We can see that the risk-budget based approach is not a single policy; it is really a policy family. Each policy instance constrains operations in its own way and affects operational behaviour and effectiveness. The question arises: for a specific mission (with all its nuances and characteristics), which of this vast family of policies will lead to the best overall results? Currently there is no way of knowing. We believe that such a policy must be found rather than specified without investigation and finding a policy would appear to be a computationally hard task. This shifts the emphasis away from specifying and refining a one-size-fits-all policy towards searching for a policy that has beneficial and acceptable outcomes from a family of policies. We believe this is entirely novel.

This problem resonates elsewhere. Current governments use macroeconomic levers such as setting interest and taxation rates to achieve overall economic goals. The economy is a complex system with emergent properties and there is often no agreement amongst economists about the consequences of particular choices of system parameters. Different countries also have different sorts of economy: there is no one policy fit for all. This is precisely the situation we are in for military operations. The goals and capabilities of organisations and missions will vary, as will capabilities and staffing characteristics. Why would we expect a one-size-fits-all security policy to satisfy our needs and allow us to make appropriate risk decisions in all cases? Parts of a given security policy may apply across settings but some notion of mission or organisational specificity needs to be taken into account. We observe that it may be difficult to determine the effect a particular economics based policy may have on attaining mission goals. (Indeed, predicting the operational effects of almost any security policy may be hard. Emergent properties are recognised as a major difficulty across the system engineering discipline.)

Similar problems appear in many other domains. Consider, for example, the proliferation of social networking websites and the privacy issues that arise with regard to the information published. Coming up with a default privacy policy to suit all users is inherently difficult; yet simply providing fine grained controls that allow users to set their preferences is neither sufficient nor realistic to solve the problem [9]. The process of specifying who is allowed to access which information can be cumbersome and becomes a usability nightmare. Consequently, users are unlikely to use these privacy controls even if they are made available to them. As discussed earlier, the task is far from being straightforward. Even if the users were dedicated in spending time to set up their custom privacy policy, they might not be adept in doing so.

1.2 Technical approach

Our approach to the above problems is a radical one. We view security policy management as a control problem. Authorisation to carry out particular actions is usually given before an action is carried out. However, operational needs may well require that local decisions be taken that are subsequently subject to review. Thus, security management may say "yes, you were in a tough situation and that was acceptable". Policy change is thus understood as a control problem, with security management giving feedback to define and redefine the acceptable envelope.

We investigate interactions between real-time decision making and security management control actions. In particular, we investigate how a specific set of decisions may be generalised into an applicable security policy. Thus, if management authorises a specific instance (either in real-time or post facto), which the underlying security policy does not, we might reasonably conclude there are many difficult but similar examples that might also be authorised. Similarly, wider constraints would seem to follow for refusals. What should those wider relaxations or restrictions be? We propose to investigate a variety of inference techniques. A developed inference system could generate policy rules and then pose interesting instances to confirm or contradict generalisation. We would envisage that there would be limits on how far or how fast that system policy can evolve without security management intervention.

Essentially, we are seeking learning techniques to partition the risk-decision space, with each partition defining an appropriate risk-related response, e.g., yes, no, yes but only if additional requirements are met. A presented vector of risk-related information associated with a request will be categorised in order to determine the appropriate response. There is considerable flexibility available in the way we may choose to recognise and codify appropriateness of yes/no/conditional yes responses. Traditionally, security policies are generally developed on the basis of human reviewable rules of one form or another. This seems rather restrictive; there are many instances where relaxing the requirement of human comprehensibility enables more effective solution. For example, bio-inspired techniques such as evolutionary algorithms (EAs) and artificial immune systems can outperform traditional techniques in many engineering domains [10, 11].

In this thesis, we investigate the potential of using EAs as the means of determining the optimal, or at least excellent, security policies for initial system configuration and also the means of adapting the security policies in the light of changing circumstances. EAs are heuristic optimisation techniques that draw loose inspiration from the natural selection process of the biological world. EAs often begin with an initial population of individuals that are randomly generated. Each individual represents a candidate solution to the problem in question. This population is then repeatedly subjected to the evolutionary process, which evaluates the fitness of each individual (i.e., how good the solution is) and selects these individuals according to their fitnesses to breed the population of the next generation. This process produces populations that are increasing better in solving the problem. This process is often iterated until the predefined maximum number of generations has been reached or a “good enough” solution has been found.

The main reason that we choose EAs is that these techniques make very weak assumptions on the solution space and have the ability to search for solutions of unknown or controlled size and shape in vast, discontinuous solution spaces. Moreover, these techniques have achieved many significant successes in their applications on many other domains, especially when the problems are computationally hard. Other data mining algorithms and heuristic search techniques are also potentially applicable.

Two EAs chosen here for investigation are Genetic Programming (GP) and Grammatical Evolution (GE). These techniques were originally proposed to evolve computer programs; security policies are essentially decision making programs. The individual representations that these techniques use fit the problem very well. The tree based individual structure in GP and the use of grammar rules to map binary individuals to programs in GE are both very flexible in presenting a program. The use of grammar rules in GE also provides the ability to constrain the search space efficiently and the advantage of decoupling the search component from the solution representation (programming language). Additionally, GP has been shown to be very effective in evolving competitive artifacts. There are currently more than forty human-competitive artifacts successfully evolved [10].

To investigate this approach, we need decision examples. Ideally, these examples should be from the real world. This could be from monitoring manual decision making or else as part of a more standard requirements elicitation activity. However, it is very difficult to obtain such examples. From a security perspective, revealing such information, to an organisation, can be very risky. (Having said that, we had tried without success to acquire such decision examples from various organisations and research collaboration partners.) Moreover, MANETs of the sorts we envisage do not really exist right now. Aiming to acquire decision examples from such environments is doomed to fail.

To overcome this problem, we generate these decision examples by running known, standard policies. Using these examples, we attempt to use EAs to infer the original policies. We start with some simple binary decision policies, e.g., MLS Bell-LaPadula policy model and budgetised MLS policy model and then continue with a multi-decision model, e.g., Fuzzy MLS policy model. We also investigate the performance of various EAs. To investigate how well EAs can mould and shape the policies to adapt with new decision criteria with new decision examples, we design a simple, yet non-trivial policy that varies with time. This time-varying policy has two purposes: generating training decision examples and serving as the benchmark against which the security policies learnt are evaluated. Lastly, we show how simulation runs can be used in place of a set of decision examples to learn the optimal security policies for a specific set of missions of concern.

1.3 Thesis hypothesis

Formally, the hypothesis of the thesis is stated as follows:

> Evolutionary algorithms (EAs) have the potential to be an effective means of determining the security policies that suit dynamic challenging environments.

By effective we mean the ability to determine the (near) optimal security policy that fits the needs of the mission in its operational environment. We attempt to examine this hypothesis from three different perspectives:

- Exploring the potential of EAs in inferring the (near) optimal static security policies from a set of decision examples.
- Exploring the potential of EAs in dynamically updating security policies with new decision examples.
- Exploring the potential of EAs in searching for the (near) optimal policies that fit a specific set of missions using simulation runs.

Our approach is radical and arguably targets systems that, in a real sense, will not exist in practice for several years yet. However, we believe it is appropriate to investigate the above in order to be prepared when such systems come on stream. We will not be able to evaluate our approach on full scale policies (because they do not yet exist). However, we do aim to establish the plausibility or, at the very least, a greater understanding of the strengths and weakness of our approach.

1.4 Thesis organisation

The subsequent chapters of this thesis are organised as follows:

Chapter 2 presents the fundamental computer security concepts that are related to this thesis. It begins with a brief introduction to security objectives and security risk analysis. It then presents the concept of dynamic coalitions, with an emphasis placed on MANETs and the challenges they impose on current security mechanisms.

Chapter 3 firstly reviews some influential security policies and models. It then presents some recently proposed risk-budget based models that aim to provide more flexibility and discusses the top-down policy hierarchical development model that enables policy composition and refinement. The chapter concludes with an identification of those research issues in security policy development that we will attempt to address in this thesis.

Chapter 4 introduces the learning techniques used in this thesis. The first part introduces EAs; it begins with a brief introduction to the common features shared among all EAs followed by the concept of multi-objective evolutionary algorithms (MOEAs). It then details two EAs used in this thesis, namely GP and GE. The second part introduces fuzzy expert systems.

Chapter 5 details the experiments in using EAs to infer static security policies from a set of training decision examples. It begins with experiments that attempt to infer some simple binary decision policies and continues with experiments that attempt to infer a more complicated multi-decision policy. Lastly, it details an experiment that demonstrates how the fuzzy set concept can be integrated into the policy inference framework to improve the performance.

Chapter 6 begins with the design of a time-varying, risk-budget based security policy model. This model is used to generate decision examples that are to be used for training as well as evaluation purposes. It then details the experiments carried out in using MOEAs to continually infer the dynamic policy from the generated decision examples, i.e., evolves and adapts a policy as new decision examples are obtained.

Chapter 7 presents the experiments in using MOEAs to discover the (near) optimal policies that fit a specific mission (or at least a specific family of missions). The experiment also shows how simulation runs can be used in place of a set of decision examples in evaluating the fitness of a policy with respect to the specified high-level objectives.

Chapter 8 concludes the thesis by evaluating the degree to which the hypothesis has been justified and outlines potential work for the future.

Chapter 2

Computer Security

Security is about the protection of assets against threats [12]. This definition implies that we need to know what assets require protection. As computer systems have evolved, the nature of specific assets and threats has changed [13]. Prior to the invention of the personal computer, computer security was mainly concerned with the protection of computer mainframes. Here, particular threats could be countered by simple physical controls. For example, storing a mainframe in a room with effective physical access controls to prevent unauthorised access.

In recent years, computer and network hardware has grown cheaper; using a computer has now become commonplace. Individuals use computers to store their private information, e.g., credit card numbers, bank account passwords, private diaries. Organisations use computers to increase their operational efficiency. There is an increasing amount of valuable information stored in computers. The sheer ubiquity of valuable information signifies the importance of security as an issue for us all.

This chapter presents the fundamental computer security concepts related to this thesis. It begins with a brief introduction to common security objectives and security risk analysis. It then presents the concept of dynamic coalitions, with an emphasis placed on MANETs and the challenges they impose on current security mechanisms.

2.1 Computer security objectives

Traditionally, the objectives of computer security are commonly summarised as confidentiality, integrity and availability; often collectively known as the C-I-A triad [13]. Over time, many security practitioners have realised the incompleteness of the triad and attempted to augment it with new objectives. These objectives include authenticity, accountability and non-repudiation. In [14], Donn B. Parker introduced the Parkerian hexad, which adds three more objectives to the C-I-A triad: possession (control); authenticity; and utility.

The following sections briefly summarise each of these objectives and their established scopes respectively. For extensive discussion, refer to [15–17].

2.1.1 Confidentiality

Confidentiality is concerned with the protection of information from unauthorised *disclosure*. In computer systems, confidentiality is about preventing unauthorised subjects from *reading* information. Confidentiality is often confused with the terms “secrecy” and “privacy”. Gollmann clarifies these terms in [15]. He views both secrecy and privacy as forms of confidentiality. Whilst privacy is concerned with the confidentiality of personal data, secrecy is concerned with the confidentiality of organisational data. For example, a privacy violation happens when an organisation shares the personal information of its customers with other organisations without the knowledge (or permission) of the customers.

Sometimes, it is necessary to protect the confidentiality of subject identities. This objective is often known as anonymity. More formally, anonymity is the state in which a subject’s true identity remains unknown by other subjects [18]. An example to show why anonymity is necessary is the traffic analysis attack [15]. The attackers can derive information such as the relationship between the parties from patterns in communication, even when messages are encrypted. In order to preserve the anonymity of the subject identities, there needs to be a property of unlinkability between identities of the participants and the communication.

2.1.2 Integrity

In computer security, integrity is concerned with the protection of assets from unauthorised *modification* [15], as opposed to "the quality of having strong moral principles" defined in the Oxford Dictionary of English [19]. In computer systems, integrity is typically about preventing unauthorised subjects from *writing* information. For this reason, integrity is sometimes perceived as the dual of confidentiality and similar techniques can be expected to be used in achieving this objective, e.g., the Biba model [20] has the mirror properties of the Bell-LaPadula model [5].

Further interpretations and constraints on what integrity implies have also been made in the literature. Clark and Wilson argued that the usage of a data modification method, which causes data loss or corruption, should not be permitted even by an authorised subject [21]. The integrity requirement is split into two parts: internal and external consistency. Internal consistency is concerned with ensuring the consistency of data representation and modification within the computer systems; external consistency is concerned with ensuring that the data reflect the real-world objects that they represent. The Clark-Wilson definition is more sophisticated and reflects the subtleties present in commercial environments. On the other hand, the definition of integrity given in the Trusted Computer System Evaluation Criteria (TCSEC) is concerned only with external consistency; integrity is defined as "the state that exists when computerised data is the same as the source documents and has not been exposed to accidental or malicious alteration or destruction" [22].

2.1.3 Availability

Availability is concerned with the likelihood of a system is able to provide some services. In particular, the availability at time t, usually denoted by $A(t)$, is the probability that the system can provide a specific service at time t under stated conditions [23].

Availability may be compromised by a variety of mechanisms. A simple example is hardware failure. Traditionally, the threat from such failure is countered by the use of redundancy [24]. Redundancy can be made in two forms: either the redundant components act as backups that are activated should one component fail, or all duplicate components run concurrently and form a voting system, in which the consensus output is the majority vote. Denial of service (DoS) attacks, which aim to make a system unavailable to the authorised users, may take many forms. At one extreme, an army of compromised hosts may be used to clog up a large network by wide-scale consumption of resources. At the other, a smart attacker may target a specific server aiming only to issue service requests at the rate they are dispatched but in a manner that keeps the service request queue full, and hence unavailable. This is usually known as a low-rate DoS attack [25].

2.1.4 Authenticity

Authenticity is concerned with the genuineness of the identity a subject claims to be. Something is said to be authentic when it really is what it claims to be. Authentication is the verification of such claims [17]. We may be interested in verifying that the user at a terminal is who he claims to be. This is personal identity authentication. Authentication is clearly a prerequisite for many other aspects of security. Access control is used to dictate the access given to subjects with regard to specific objects. However, it makes the assumption that the subject in question really is the person concerned or acts legitimately on that person's behalf, e.g., a process started by that user.

There are a great many means of authenticating the identity of a person. These can be loosely categorised into three groups as follows:

1. Something the user has, e.g., a token card.

2. Something the user knows, e.g., a password, a pin, a signature.

3. Something the user is (giving rise to biometrics such as fingerprints, iris patterns, and various behavioural characteristics such as dynamic signature properties).

An example of authenticity violation is an attacker logging in as an ordinary user using a stolen password.

In many security protocols, received messages may seem to be recently created. However, we know that messages can be recorded and replayed and thus it is often required to verify any such claims to recency. This is a form of message authentication. Often, we may also wish to verify the sender identity of a received message, e.g., the sender of an email that requests a bank statement.

2.1.5 Accountability and non-repudiation

The security objectives discussed so far have sought to prevent unwanted events from happening. What if these preventions fail? Accountability attempts to answer this by ensuring the actions that affect security can be traced to the responsible subject [15]. In other words, accountability attempts to establish the links between the subjects and the actions made. This often conflicts with anonymity that strives to unlink them. A common way to achieve accountability is to keep a secure audit trail on the systems. Illicit modification or deletion of an audit trail would clearly compromise accountability. A DoS attack on the audit server provides an alternative and possibly easier way to achieve the same goal.

Non-repudiation is a strong form of accountability. Non-repudiation is concerned with the ability to ensure that a subject is unable to deny carrying out the action [15]. This objective is commonly achieved with the use of digital signatures. In signing a piece of data with a private key, an unforgettable binding is established between the subject and the data. Disclosure of a private key would clearly compromise any claims to legitimacy of binding [15]. Thus, users must keep their private keys secure.

2.1.6 Summary

The security objectives discussed in the section can be summarised as follows:

1. Confidentiality — Prevention of unauthorised disclosure of information.

2. Integrity — Prevention of unauthorised modification of information.

3. Availability — Prevention of the DoS.

4. Authenticity — Verification of identity one claims to be.

5. Accountability and non-repudiation — Prevention of the denial of actions made.

2.2 Security risk analysis

Security risk analysis is the process of ensuring that the security of a system is commensurate with the risk it is exposed to. All protection measures come at a price. Security risk analysis provides a means to justify the tradeoff between cost and benefit for the security controls implemented. Despite the various methodologies in conducting security risk analysis and some of them being tailored to a particular discussion, they all share a common framework composed of the following steps: assets, vulnerabilities and threats identification, risk assessment, selection of control, and re-evaluation.

2.2.1 Assets, vulnerabilities and threats identification

Assets are resources that have values in a system [13]. Assets in a computer system can be mainly categorised into three groups: hardware, software and information. At times, the workforce and the reputation of a company are considered as part of the assets [15]. To do risk assessment, all assets are first identified with their values evaluated. Whilst the values of tangible assets are easy to quantify by considering the monetary replacement cost, the values of intangible assets are difficult to estimate. For example, the loss of confidential information on suppliers and clients may affect the reputation of the company. In addition, any damage to the assets in the above categories can cause damage to the quality of service. One possible way to estimate the values of these assets is based upon their importance [26].

Vulnerabilities are the weaknesses of a system. Attackers attempt to discover the vulnerabilities of a system in order to cause damage to assets, either accidentally or intentionally [15]. Vulnerabilities can exist at different levels and places in a computer system, e.g., operational environments, operating systems, application software, networks, communication media and operational practices.

Threats are the potential actions that can be used by attackers to exploit vulnerabilities in order to cause damage to assets [1]. Threats are caused by threat agents, which can be both internal and external to the system. Examples of threat agents may be hackers, system administrators or viruses that exploit bugs in a software to launch attacks on a system. In the literature, the term "threat" is often used where the term "threat agent" should be. One example is the definition of threat as "the party with the capabilities and intentions to exploit a vulnerability in an asset" [27].

The relationship between these terms is best illustrated by the following example. In a computer network, a possible vulnerability is the use of a default password at the network router (asset). A hacker (threat agent) can exploit this vulnerability to take control over the router and launch a DoS attack to prevent authorised computers from connecting to the network.

2.2.2 Risk assessment

The definition of risk varies considerably in the literature depending on the domain in which it is considered for. For example, risk can be the standard deviation on the return of an investment in finance [28], or a function on the amount of loss and the probability of the loss [2, 29]. Nevertheless, there is a common theme behind these definitions. Risk is always related to expected loss, which can be caused by an action and the estimated probability of such loss arising.

In security, risk is defined as a function of the value of assets, vulnerabilities and threats [15]. This definition is coherent with the engineering definition of risk by considering the amount of loss as a function of the value of assets and vulnerabilities in the system. Based on this definition, the risk assessment of a computer system can be carried out quantitatively or qualitatively [15].

In a quantitative assessment, the values are calculated using various mathematical theories and formulae [30]. For example, risk can be calculated based on the monetary replacement values of assets and the probabilities of threats happening. The advantage of this analysis is that it provides a precise numerical risk value, which is useful for cost-benefit analysis of recommended controls [30]. However, the precise meaning that a given numerical risk value represents can become unclear, e.g., a high risk value can be due to the high value of the asset, the high probability of threats happening, or both factors. This may cause problems in selecting suitable controls to protect the system assets because different assets may require different protection mechanisms.

In contrast, a qualitative assessment uses descriptive variables to represent risk [30]. For example, each asset is given a value on a scale of cheap, medium and expensive; criticality of vulnerabilities is given a value on a scale of very low, low, medium, high and very high; and each threat is given a value on a scale of very low chance, low chance, medium chance, high chance and very high chance. The mapping of these values to the risk can be obtained by using a mapping table based on the advices of security experts [26]. There are also other qualitative analysis techniques, including scenario analysis and questionnaires [15]. The advantages and disadvantages of using a qualitative analysis are more or less the mirror of using a quantitative approach. This analysis provides a means to identify the vulnerabilities of the systems in a relatively short time [15]. However, the cost-benefit analysis of recommended controls becomes difficult in the absence of a precise numerical risk value [15].

2.2.3 Selection of controls

Controls, also known as countermeasures, are the ways to protect a system against threats. The controls selected must be commensurate with the risk identified. Controls for the computer systems can be categorised into three types: administrative, physical and logical.

Administrative controls are concerned with the relationship between security and human factors [31]. Administrative controls specify how a system can be used. Examples of administrative controls include organisational security policies, user registration processes, business continuity plans and disaster recovery plans. For example, the computing service in a university defines the security policy that students must agree and abide by. It is often the case that this high level security policy, defined in administrative control documents, forms the basis of the selection of logical and physical controls. In other words, the logical and physical controls implement and manifest the administrative controls.

Physical controls protect the physical hardware of computer systems from physical and environmental threats. Some examples include locks, closed circuit surveillance cameras (CCTV) or security guards (protection from unauthorised accesses), cooling systems (protection from heat) and backup sites (protection from natural disasters).

Logical controls protect computer systems using software and data measures. Some examples include data encryption, access controls, firewalls and intrusion detection systems. In recent years, logical controls have received much attention from the security community and have achieved significant advancement. Consequently, the knowledge gained in logical controls has also been transferred to protect and to improve physical controls, e.g., the use of PINs in conjunction with door entry cards.

Security is only as strong as the weakest link [32]. In practice, all three types of controls have to be implemented and balanced in order to achieve security objectives of concern. For example, for availability, strong physical controls such as a reliable cooling system and backup site are needed to protect the physical hardware of a computer system from physical threats. At the same time, strong logical controls such as a strong user authentication process and a good access control policy are also needed to protect the system from unauthorised accesses to the services provided. The lack of either control can easily result in the system becoming unavailable.

2.2.4 Re-evaluation

A detailed security risk analysis on large computer systems is not feasible for every organisation as it requires significant resources (time and cost) [1]. Furthermore, new threats and vulnerabilities emerge each day because the operational environment is constantly changing. Therefore, security risk analysis is an ongoing iterative process and must be indefinitely repeated.

2.2.5 Summary

A conceptual model that summarises a security risk analysis process is shown in Figure 2.1. In a security risk analysis process, the system owners attempt to quantify the risk of the system exposed to by analysing vulnerabilities of the system and identifying the threats to system. Based on the risk, appropriate security controls are then selected to protect the system. The ultimate objective of the process is to ensure that the security controls of a system implemented are commensurate with the risk that the system is exposed to.

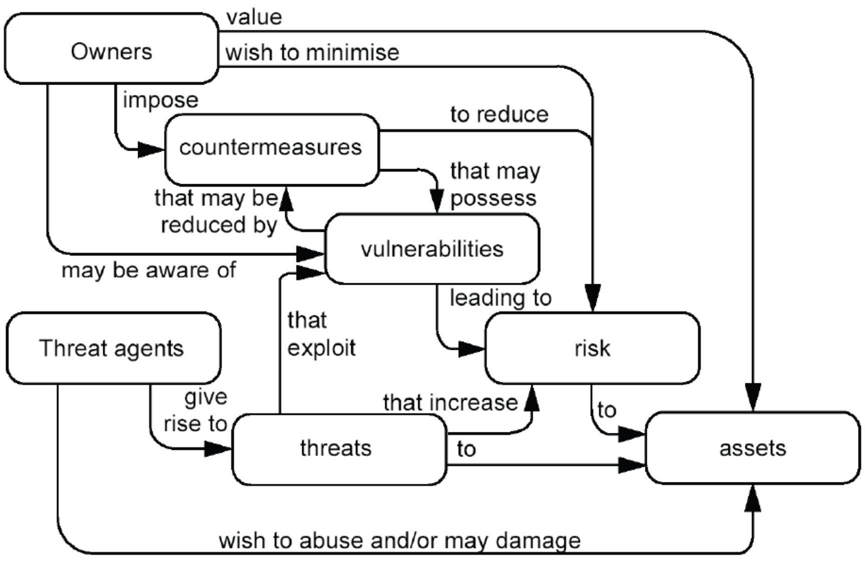


Figure 2.1: The conceptual model of a security risk analysis process [1].

2.3 Dynamic coalitions and MANETs

A coalition is defined as a "temporary alliance for combined action, especially of political parties forming a government" in the Oxford Dictionary of English [19]. This definition underpins two important characteristics of a coalition. Firstly, a coalition is temporary; the alliance between parties is not permanent and will cease to exist in the future. How long the coalition lasts depends on the type of coalition. Secondly, an alliance is formed to achieve a common objective, which may be difficult or even impossible to achieve alone by each individual party. This implies that sharing of resources such as objects, applications and services is an integral part of a coalition. Without common objectives, each party may not be willing to share their resources [33].

A dynamic coalition is a coalition that allows parties to join or leave during the lifetime of the coalition. Some examples of dynamic coalitions are as follows [33, 34]:

1. In a war, two or more countries may come together in an alliance to strengthen their forces. These countries may decide to share some classified information such as locations of forces and bases to increase the efficiency

of the operation. However, a friend today might become a foe tomorrow; an alliance member can change sides and become an opponent at any point in time. The opposite can also occur.

2. A real-time systems research group in a university discovers a new highly efficient scheduling algorithm and wishes to form an alliance with a private automobile manufacturer and an embedded chip company for evaluating the performance of the algorithm in the real world. Given the interest, all three parties come together, form an alliance and share all research data generated. After the coalition is set up, other organisations may decide to join in or some of the initial members may decide to leave.

3. In the aftermath of an earthquake, the police department, the military forces as well as voluntary organisations such as the Red Cross come together and form a rescue operation alliance. The police moves refugees to a safe place, whilst the Red Cross provides the medical and food aid. The military forces provide transport to make the rescue operation more efficient. Eventually, there is a recovery phase and basic infrastructure is rebuilt to get the region back to normality.

4. MANETs are a type of network that can be rapidly deployed without relying on existing infrastructure. The nodes in MANETs can dynamically join and leave the network, often without warning, and possibly without disruption to communication of other nodes, refer to Section 2.3.1.

Resource sharing is at the heart of every dynamic coalition. Each party in a coalition hesitates to share its resources including information with other parties in order to minimise its risk, yet sharing is necessary to achieve the common objective of the coalition. This problem is commonly known as the Dynamic Coalition Problem [35]; having well designed access control policies and mechanisms are vital in solving this problem.

2.3.1 MANETs

Wireless networks have grown and changed rapidly in the last decade. Wireless networks can be categorised into two groups: infrastructure based networks and ad-hoc networks. In infrastructure based networks, there are some preinstalled equipments, i.e., base stations to which all the mobile nodes are connected. All communication between the nodes passes through base stations. A base station may also serve as the gateway of a wireless network to a wired network. When a mobile node moves out of the range of a base station into the range of another, a hand off process is executed automatically and the mobile node continues its connection seamlessly with the network [36]. Mobile phone service networks are good examples of this type of network.

In ad-hoc (infrastructureless) networks, there is no preinstalled infrastructure that the nodes can rely on to connect to each other. The nodes in an ad-hoc network dynamically connect to form a multi-hop network. Each node plays the role of a router, discovering the route and forwarding the data for other nodes dynamically [36]. Ad-hoc networks are also self-configuring and self-organising networks. Self-configuring in the sense that an ad-hoc network can be formed on the fly; self-organising in the sense that the network can change based on its needs, either by partitioning or merging the network with few administrative actions. The two common types of ad-hoc networks are MANETs and sensor networks.

MANETs are a subset of ad-hoc networks with highly dynamic network topologies [37]. Historically, MANETs have been pioneered by the military. The first two projects on MANETs were the Public Radio Network (PRNet) [38] and its follow up project, Survivable Radio Networks (SURAN) [39], both funded by Defence Advanced Research Projects Agency (DARPA)[1]. Most existing non-military MANETs have been developed in academic environments. The notion of MANETs in the commercial world can be traced to the emergence and success of the IEEE 802.11 Wireless Local Area Networks (WLANs) and Bluetooth technologies in the 1990s. Due to the popularity of MANETs, the term "ad-hoc networks" is often used interchangeably with "mobile ad-hoc networks" in the literature.

Sensor networks are high density networks with a large number of sensor nodes deployed in an area to monitor phenomena of interest. The discussion on sensor

[1]DARPA is an agency of the US Department of Defence that is responsible for the development of new technology for use by the military.

networks is beyond the scope of this thesis. For a comprehensive survey on sensor networks, refer to [40].

2.3.2 Security challenges of MANETs

The flexibility that MANETs offer comes at a price. The infrastructureless nature presents many security challenges that are specific to MANETs as follows [37, 41, 42]:

1. Lack of trusted entities — Lack of infrastructure is a fundamental characteristic of MANETs. The lack of a trusted, centralised entity in MANETs requires network administrative tasks to be distributed among the nodes in the networks. This results in increased security risk as there are more possible access points for intrusion. Moreover, many of the existing security protocols, authentication and access control mechanisms rely on the existence of a trusted, centralised entity, e.g., the public key infrastructure (PKI) requires a centralised trusted certification authority (CA).

2. Routing attacks — Nodes in MANETs organise themselves to communicate with their neighbours in such a way as to provide connectivity across the networks. As the nodes are mobile and have the freedom to move in an arbitrary manner, the network topology changes frequently in an unpredictable fashion. Consequently, communication routes between nodes also change and network partitioning may happen if there is no overlap in network coverage of two or more sets of nodes. The routing algorithm has to be highly adaptive and robust to accommodate these frequent changes. There are many possible attacks on routing identified in the literature. In the simplest case, the routing table in the nodes can be directly modified once they are captured by adversaries. Packets can also be maliciously created, modified or dropped to change the routing table. Additionally, attacks on routing can be launched by making the nodes inactive or by making them behave selfishly (using services but not cooperating in routing tasks).

3. Resource attacks — Mobile nodes have constraints on their resources, in terms of power sources, processing power and network bandwidth. The effi-

ciency in using these resources is an important factor in designing MANETs. Often, the nodes in MANETs are allowed to switch themselves into a sleep mode to conserve energy. However, this in turn leads to the routing problem mentioned earlier. In contrast, "sleep deprivation" is another type of attack that aims to exhaust the power resource of the nodes by keeping the nodes active at all times. Furthermore, the bandwidth of wireless links is significantly less compared to wired links due to noise and interference. Consequently, there is a constraint on the amount of data that can be transmitted at one time in the networks.

4. Incompatibility of traditional cryptographic techniques — Many traditional cryptographic techniques cannot be directly implemented in MANETs for two main reasons. Firstly, many of these techniques require a centralised entity, which is not present in MANETs. Indeed, it is often the case that no node in the network may be assumed to be fully trustworthy because of the hostile operational environment. Secondly, many traditional cryptographic techniques may be computationally intensive, yet the nodes in MANETs often only have limited computational and power resources.

5. Inherent problems in wireless communication — MANETs inherit all the security problems of wireless networks. The wireless communication medium is less reliable than the wired medium. It is necessary for the networks to be able to distinguish the variation in physical link performance and the possible forms of malicious attacks, e.g., DoS attacks. These attacks can happen at various network layers. Additionally, wireless communication is broadcast in open air and no physical security protection can be used to protect the communication channels. It is necessary to assume that adversaries can eavesdrop and possibly perform some interpretations on the transmitted signals, e.g., traffic analysis. The broadcast signal can also be used by adversaries to determine the location of the networks/nodes.

6. Operational environments — MANETs are often deployed in risky and hostile environments such as battlefields. Therefore, it is safe to assume that the networks may face more challenging security attacks than conventional

networks. Attackers can, for example, capture the nodes in MANETs and use them to launch internal attacks.

7. Security policy issues — The dynamic nature of the networks can cause the risk factors to be constantly changing. Current static security policies that consist of static rules are unable to cope with this. For example, the risk that a node is exposed to may change depending on the operational environment (e.g., time and location) and the remaining resources (e.g., network bandwidth and battery power) of the node.

2.4 Conclusions

This chapter presents a brief overview of computer security. It discusses some major security objectives, namely confidentiality, integrity, availability, authenticity and accountability. It then discusses security risk analysis, which involves the process of identifying assets and their vulnerabilities to threats, the risk assessment process, and the selection of security controls. Next, it presents the concept of dynamic coalitions and MANETs. Lastly, it presents a review on the limitations of current security mechanisms in relation to MANETs.

Chapter 3

Security Policy Models

A security policy is

> a set of rules (or principles) that direct how a system (or an organisation) provides security services to protect sensitive and critical system resources [43].

A security policy must therefore specify *all* necessary control measures for ensuring system security, including how authentication should be done and what responses should be made to a security violation, etc.

Access control specification is typically a major component of a security policy. Access control protects the resources of a system (objects) against unauthorised access by restricting the use of system resources only to the authorised users (subjects) according to the security policy of the system [43]. Access can be in many modes; the common ones include read, write, append and execute. The access mode that a subject has on the objects in the system is known as the “privilege” the subject has. The set of high-level rules that specifies which accesses are to be authorised and which are not is known as the access control policy [44]. The study of access control policies has resulted in various useful models. Most of these models are formal, i.e., formal analysis can be carried out to prove the models are secure with respect to the security objectives concerned. There are also some models consisting of informal high-level principles such as the Clark-Wilson model [21].

This chapter first reviews some traditional influential security policies and models in the literature. It then presents some recently proposed risk-budget based models that aim to provide more flexibility. Next, it introduces the top-down policy hierarchical model that enables policy composition and refinement, with an emphasis on the problems encountered in the refinement and conflict analysis processes. Lastly, it summarises the current state of the art in security policy development and reiterates the research objectives of the thesis.

3.1 Mandatory access control policy models

The Multi-Level Security (MLS) system was first developed in the defence community as a means to manage information with different sensitivities. In the system each object such as a file is given a classification label that represents its sensitivity and each user is assigned a clearance level which is on the same scale as the classification. The clearance level a user has depends on the user's background, e.g., rank and the results of checks into the user's behaviour and lifestyle. Essentially, these factors are used to establish the trustworthiness of the user and the information the user needs to know. The clearance level of the user is then subsequently determined and granted. Mandatory Access Control (MAC) policies enforce access control to objects by comparing the classification labels of the information (objects) with the clearance levels of the users (subjects) [22]. This is typified by the Bell-LaPadula Model [5].

3.1.1 Bell-LaPadula model

The Bell-LaPadula model is concerned with the confidentiality of objects in an information system. In this model, each entity in the system is either a subject or an object. Each entity is associated with a security label of the form ⟨classification level, category set⟩. The set of classification levels consists of names that form a dominance ($>$) relation based on the sensitivity or the clearance. An example of classification levels can be {top secret, secret, confidential, unclassified}, where top secret $>$ secret $>$ confidential $>$ unclassified. Here $>$ means "is more sensitive than". The set of categories contains names that form compartments which provide a means to the creators of the objects to control and restrict the distribution of the objects. An example of categories can be {investment banking, equity, technology} in a financial institute.

With such setting, a dominance, $\succeq$ relation can be defined on security labels as follows:

Dominance, $\succeq$. *Let* $\langle s_a, c_a \rangle$ *and* $\langle s_b, c_b \rangle$ *be the security labels of* a *and* b*,* $\text{label}(a) \succeq \text{label}(b) \Leftrightarrow (s_a \geq s_b) \wedge (c_a \supseteq c_b)$.

This relation forms a partial order set (poset) on the security labels. Two properties are defined to ensure the system remains secure with respect to the confidentiality of the objects:

1. The simple security property (no read up) — no subject can read any object with a higher security label than its own security label. This can be rewritten in a non-negated form as $\forall s \in \mathit{Subject}, o \in \mathit{Object} : s$ can read $o \Leftrightarrow \text{label}(s) \succeq \text{label}(o)$.

2. The *-property (no write down) — no subject can write to any object with a lower security label than its own. Formally, this can be written as $\forall s \in \mathit{Subject}, o \in \mathit{Object} : s$ can write $o \Leftrightarrow \text{label}(s) \preceq \text{label}(o)$.

In other words, the Bell-LaPadula model restricts information to flow from low confidentiality to equal or higher confidentiality. Later, a tranquillity property is added to explicitly state that security labels of the entities in the system cannot change during system operation to rebut some criticisms received in [45, 46]. (If security labels can change arbitrarily under system operation, then clearly security can be compromised. Consider a system in which the security labels of all subjects are changed to the maximum possible levels allowing all files to be read by everyone.)

Whilst MAC is often associated with the Bell-LaPadula model, they are not the same. MAC really means that the security policy which governs the system is enforced by the system itself, as implied by the term "mandatory". The subjects cannot manipulate access control attributes of the objects they own at their own discretion. The Bell-LaPadula model is just an instance of MAC.

3.1.2 Biba model

A model that is closely related to the Bell-LaPadula model is the Biba model [20], which is concerned with the integrity of the objects in a system. It is essentially the inverse model of the Bell-LaPadula model, i.e., the information is restricted to flow from high integrity to equal or lower integrity. Two properties are defined to ensure integrity of the system:

1. The simple security property (no write up) — $\forall s \in \mathit{Subject}, o \in \mathit{Object}$: s can write $o \Leftrightarrow \text{label}(s) \succeq \text{label}(o)$.

2. The *-property (no read down) — $\forall s \in \mathit{Subject}, o \in \mathit{Object}$: s can read $o \Leftrightarrow \text{label}(s) \preceq \text{label}(o)$.

The Biba model also introduced two new concepts: the dynamic security label and the invocation operation. The dynamic security label relaxes the constraint of static security classification in the Bell-LaPadula model. An access to a low integrity object by a subject operating at high integrity causes the operation of the subject to be dynamically downgraded to low integrity. The new invocation operation enables subjects to invoke other subjects (probably software processes) to access objects. To remain consistent, the invocation property is defined such that subjects are only allowed to invoke other subjects with lower or equal security labels[1].

3.1.3 Chinese wall model

The security models introduced so far are inspired from military applications. The Chinese wall model [47] is a commercially inspired model which is concerned with confidentiality by ensuring that information flow in the system does not have any conflict of interest. A classic scenario example is the services offered by a financial institution to different clients, some of which are competitors to

[1]The ring property is an alternative property defined such that there is no restriction set for all read accesses, i.e., any subject can read any object regardless of their security labels, but a subject s_1 can write an object $o_1 \Leftrightarrow \text{label}(s_1) \succeq \text{label}(o_1)$ and a subject s_1 can invoke a subject $s_2 \Leftrightarrow \text{label}(s_1) \preceq \text{label}(s_2)$. However this property is inconsistent with the other properties defined here. One must choose which is more appropriate to use depending on the application.

the others. A conflict of interest can arise when an analyst in the institution is involved with two companies in the same market because the insider knowledge that the analyst gains in one company may result in an unfair treatment of the other and vice versa.

The Chinese wall model groups all objects related to a company in a company dataset. Datasets with conflicts of interest are grouped under the same conflict class. If necessary, company datasets can be "sanitised" into a dataset containing no critical information. No one should be authorised to access more than one "unsanitised" dataset from the same conflict class. As the conflict depends not only on the current object to be accessed but also all other previously accessed objects, the access history of each subject must be kept.

Two properties are defined to ensure valid access control:

1. The simple security property — a subject can access an object only if the object is within a currently held dataset, or an entirely different conflict class. However, this property in itself does not prevent indirect information leakage. A subject can read information from one dataset and write in another dataset for other subjects to read. Therefore the *-property is defined to prevent this kind of violation.

2. The *-property — a subject can write an object only if the subject does not have read access to any other company datasets which are not "sanitised". This property ensures that the unsanitised information flow is restricted within its own company dataset whereas the sanitised information flow is not restricted throughout the system.

In contrast to the Bell-LaPadula model which assumes that access rights are static, the Chinese wall model assumes that access rights are dynamic and therefore have to be reexamined during all state transitions. The identification of the importance of access history also proliferates the research into history based access control.

3.1.4 Clark-Wilson model

The Clark-Wilson model [21] is a commercially inspired model which is concerned with integrity. Unlike other models, the emphasis of this model is not on what a subject can access, but how the access is done. The core of the model is based on the two well established principles in the commercial world: well-formed transactions and separation of duty.

A well-formed transaction is a series of operations which changes the data in the system from one valid state to another. This is to preserve the internal consistency of data in the system. The separation of duty principle requires that the entity that certifies a transaction and the entity that executes the transaction be different. Provided that these entities do not conspire, this principle should preserve the external consistency of the system, i.e., the data in the system reflects the data it represents in the real world.

The objects in the Clark-Wilson model are divided into either constrained data items (CDIs) or unconstrained data items (UDIs). Five certification rules and four enforcement rules are defined to constrain how to validate integrity of CDIs, how and who can change CDIs and how to change UDIs to CDIs.

3.2 Discretionary access control policy models

Unlike MAC, Discretionary Access Control (DAC) policies restrict access to objects based on the identity of the subjects and/or the groups to which the subjects belong. The resource owners can delegate the access privileges to other subjects at their own discretion; hence the name.

DAC policies can be represented using an access control matrix [48]. An access control matrix is a two dimensional matrix, in which the rows correspond to subjects and the columns correspond to objects. An element in the matrix specifies the privileges a subject has on an object. Let r, w and x represent read, write and execute access respectively. Then, an example access control matrix is depicted in Figure 3.1.

Whilst the access matrix provides a convenient way of expressing DAC policies, it is likely to be too large and also too sparse to be stored efficiently in

	File 1	File 2	Device
Alice	rwx	rwx	rwx
Bob	r	r	rwx
Charles	rw	rw	$-$

Figure 3.1: An access control matrix example.

practice. Consequently, it is usually stored either by columns or rows, resulting in access control lists (ACLs) and capability lists respectively.

ACLs store the access matrix by columns. Each entry in an ACL stores the access privileges a subject has on the objects in the system. This object-centred decoupling makes ACLs suitable for operating environments where the protection is central to the objects, e.g., operating systems. Conversely, if the number of subjects in an operating environment is large and constantly changing, ACLs become less suitable. The way ACLs store information is inefficient when one needs to find all the objects that a particular subject has permission to access, since a check on the ACLs of all objects in the system is required.

Capability lists store the access matrix by rows. The strengths and weaknesses of capability lists are more or less the opposite of ACLs. The decoupling is subject-centred; it is easy to check what privileges a subject has but it is tedious to check the privileges granted on a particular object.

3.3 Role based access control policy models

DAC policy models, however, pose several challenges to the privilege administrations, especially when the information system becomes large. An interesting observation is that the objects in the system are often not owned by the users themselves, but the organisations they work for. The access control decisions to the objects are often determined by the responsibilities of the roles the users hold [49]. This gives rise to the role based access control (RBAC) in which the access privileges in the system are defined in terms of the role a subject has, rather than the identity of the subject itself [49].

The basic components of RBAC are: users, permissions, roles and sessions. Two relations are defined: user assignment (UA) that associates users with roles, and permission assignment (PA) that associates roles with permissions. Both relationships are many-to-many mappings. Permissions are granted to users through roles. At any time, a user can choose to activate a subset of roles that he has been assigned; the permissions then available to the user are those associated with the roles activated in that session.

There have been many extensions made to RBAC. The focus of discussion here is on the American National Institute of Science and Technology (NIST) RBAC reference model [50] which has later been revised to become the American National Standards Institute (ANSI) standard (ANSI/INCITS 359-2004) [51].

The ANSI RBAC reference model consists of four components: Core RBAC, Hierarchical RBAC, Static Separation of Duty Relations and Dynamic Separation of Duty Relations. The Core RBAC defines the fundamental elements in a RBAC system. These include all the functional capabilities of RBAC presented so far, i.e., user assignment and permission assignment relations, session as well as support for user-role review (the ability to determine which role is given to a user and vice versa).

Hierarchical RBAC introduces the concept of role hierarchies, which allows roles to inherit the permissions of other roles, e.g., a senior role inherits all the permissions of a junior role. This often improves administrative efficiency through the reduction in the number of permission mappings. However, the permission inheritance based on the seniority hierarchy in the real world is not always suitable. In [52], Moffett presented some examples of how this can be in conflict with some control principles such as the separation of duty principle, decentralisation, supervision and review. Whilst various ad-hoc methods such as the use of private roles [53] or exceptions [54] can be introduced to overcome this problem, these methods defeat the original intent of improving the administrative efficiency.

Static Separation of Duty Relations introduces the concept of constraints on user assignments. As this concept may conflict with the concept of role hierarchies, it is specified in both the presence and absence of the concept of role hierarchies. Dynamic Separation of Duty Relations introduce the concept of constraints on the role activation in a session for users.

A common alternative reviewed in the literature is the original NIST standard which defines four conceptual models: Flat RBAC, Hierarchical RBAC, Constrained RBAC and Symmetric RBAC. Flat RBAC is essentially the same as the Core RBAC component; Hierarchical RBAC is Flat RBAC with role hierarchy support; Constrained RBAC is Hierarchical RBAC with constraints on the user assignments and role activation in a session for users; Symmetric RBAC is Constrained RBAC with support for permission-role review. In the ANSI standard specification, the support for permission-role review is left as an optional feature because it is intrinsically difficult to implement in large distributed systems [50].

The main advantage of RBAC is that it eases the administration task [49, 55]. Once the role-permission mapping is established, it is likely to remain constant. The administrative task of assigning roles to users is very much easier and less prone to error as compared to assigning permissions to the users directly.

There have been many other extensions proposed for RBAC. The extension to the group or team concept is a common one. In the same way that roles are used to group privileges, users can be grouped based upon the group or team they belong to. Each team then is associated with roles. Team based access control [56] and coalition based access control [57] are examples of this. Temporal RBAC (TRBAC) is proposed in [58] to introduce the concept of temporal constraints to RBAC. It allows the use of temporal constraints to specify the periodic role activation and also the dependencies between the role activation via role trigger mechanisms.

3.4 Flexible access control policy models

Whilst DAC and RBAC policy models provide better flexibility through permission delegation and role abstraction, the permission assignments to the users remain static. In practice, the permission assignments and the protection requirements of the objects may change over time. The task of administering the access control policy can easily become unmanageable, especially when the operational environment is highly dynamic.

Here we review some access control models that aim to provide more flexibility, namely the Fuzzy MLS model introduced in [2, 29] and the economics based models introduced in [8]. The intuition of these models is based upon the observation that access control is governed by the risk incurred and the benefit an organisation can gain from the access. An access is authorised only if the benefit outweighs the risk incurred. The risk-benefit tradeoff assessment of the models presented so far is implicitly and statically encoded in the models. These two models advocate the use of an explicit model to dynamically estimate the risk to make better informed decisions.

3.4.1 Fuzzy MLS model

The Fuzzy MLS model is an adaptive extension of the read access aspect of the Bell-LaPadula model. In the latter model, a subject can read an object if and only if the security label of the subject dominates the security label of the object, i.e., the sensitivity level of the subject (*sl*) is greater than the sensitivity level of the object (*ol*) and the category set of the subject (*sc*) is a superset of the category set of the object (*oc*). This can be interpreted informally as a subject can read an object if and only if the subject is trustworthy enough and has the legitimate "need-to-know" to access the object. The policy encoded in this model essentially divides each access as either having an acceptable amount of risk or an unacceptable amount of risk; only the accesses with acceptable amount of risk are authorised [2].

The Fuzzy MLS model continues to employ this risk based rationale, but it changes the access control from a risk avoidance system (inherent in the binary decision-making process) to a risk management system by computing the "quantified risk" estimated by the "gap" between the security labels of the subject and object.

The model consists of a risk scale that is divided into three parts: top, middle and bottom, as shown in Figure 3.2. Each access is mapped to a point on the scale. An access that is mapped to a point in the top region is denied because the risk is too high; an access that is mapped to a point in the bottom region is authorised. The middle region is further divided into multiple risk bands such that each band is associated with a risk mitigation measure; an access that is mapped to this region is authorised only if the associated risk mitigation measure can be applied to reduce the risk level to the bottom region. There is a change from a binary model of risk (acceptable/unacceptable) to one that embraces a more refined and continuous assessment of risk. The new model explicitly acknowledges that a risk decision need not be immediately clear-cut and the description “Fuzzy MLS” [2] is intended in part to convey this. (There is a vague analogy with the way the fuzzy logic replaces the classically clear-cut notion of set membership with one based on probabilities or continuous distributions. However, anyone expecting a more concrete inspiration from fuzzy logic will be disappointed.)

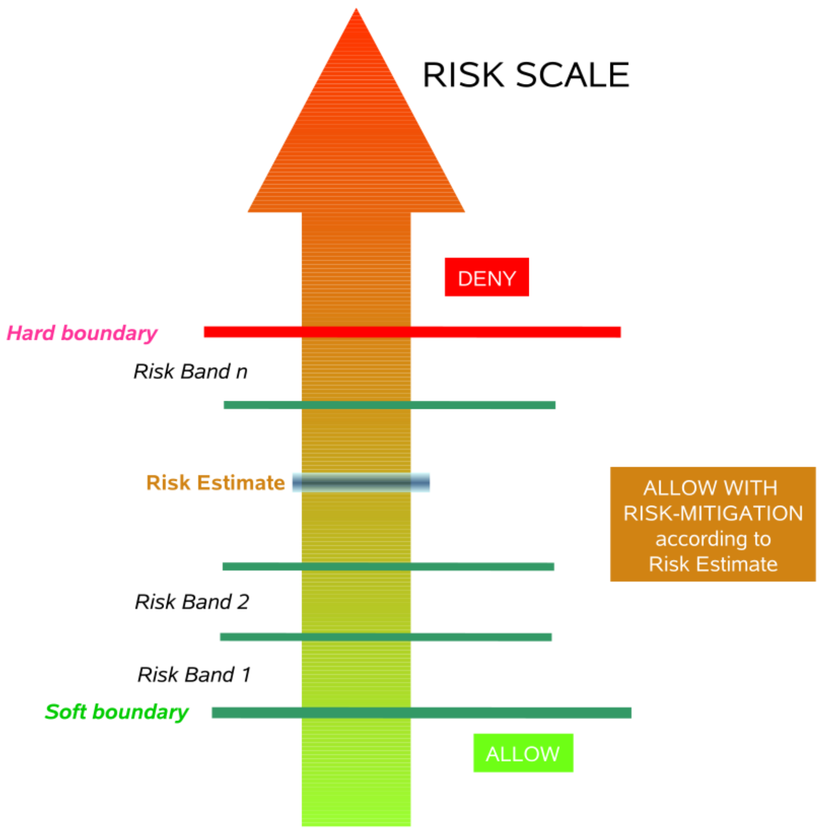


Figure 3.2: The Fuzzy MLS model [2].

Two alternative risk management systems, the credit card system and economics based system, are also discussed in [2]. In the credit card system, each subject is given a risk credit. When a subject makes an exceptional access, the difference between the risk and the soft boundary will be charged to the risk credit of the user. Periodically, the return on investment (ROI) of each user is evaluated to determine the future risk credit of the user, based on the risk credit charged and the results delivered by the user. Activities can also be logged to periodically refine the access control policy. This provides a way of governing the long-term behaviours of users, yet still maintaining sufficient flexibility as provided by the use of risk credit. In the economics based system, the notion of pseudo-currency is introduced in place of risk credit. Each user is given an amount of pseudo-currency which can be used to purchase information accesses. Riskier accesses cost more. Similar to the credit card system, the amount of pseudo-currency allocated to a user in the future depends on the ROI of the user. This is very similar to the economics based models that will be reviewed in Section 3.4.2.

Risk computation

The Fuzzy MLS model defines risk of a read access as the expected value of damage caused by unauthorised disclosure of information:

$$\text{risk} = (\text{value of damage, } V) \times (\text{probability of incurring the damage, } P) \quad (3.1)$$

The value of V can be estimated from ol. As ol typically corresponds to the order of damage, V is defined as an exponential function of ol:

$$V = a^{ol}, a > 1 \quad (3.2)$$

The value of P can be estimated by quantifying two "gaps": one between the sensitivity levels of the subject and the object, and the other between the category sets of the subject and the object. In the Bell-LaPadula model, P can be viewed as:

$$P = P_1 + P_2 - P_1 P_2 \quad (3.3)$$

where P_1 and P_2 are:

$$P_1 = \begin{cases} 0 & \text{if } sl \geq ol \\ 1 & \text{otherwise} \end{cases}$$

$$P_2 = \begin{cases} 0 & \text{if } sc \supseteq oc \\ 1 & \text{otherwise} \end{cases}$$

The Fuzzy MLS model continues to use (3.3) to estimate these probabilities, but allows these probabilities to take real values in $[0, 1]$ instead of only being expressed as binary values.

To determine the probabilities P, P_1 and P_2 precisely is impossible as no one knows the future for certain. However, risk indices can be formulated by having an assessment on the likelihood of misuse in different access operations. The higher the likelihood of misuse, the higher the risk index. Thereafter, these risk indices can be mapped onto probabilities based on previous experiences and intuitions. The use of risk indices is doubly useful. Firstly, it removes the constraints imposed by the probability theory, e.g., the sum of probabilities of all possible outcomes must be equal to one. This provides a more fine-grained view on risk. Secondly, it makes the fine tuning task easier in the future. Risk indices, once determined, can be kept fixed; only the mapping function is required to be tuned over time.

P_1 is defined as the probability of a subject failing to resist the temptation of unauthorised information disclosure. The Bell-LaPadula model can be viewed as taking a binary view on temptation: temptation happens if the sensitivity level of subject (sl) is less than the sensitivity level of the object (ol) and information disclosure always happens $(P_1 = 1)$ if temptation exists.

To formulate P_1 for Fuzzy MLS Model, the temptation index TI is first derived. There are many different ways to relate TI to sl and ol, but it would seem intuitive that the relation must satisfy the following properties [29]:

- As the sensitivity level of an object increases, the temptation increases.
- As the sensitivity level of a subject increases, the temptation decreases.
- Every subject faces temptation, i.e., there is no 100% trustworthy subject.
- TI is biased towards more sensitive objects.

A simple formula [2] that satisfies these properties is:

$$TI(sl, ol) = \frac{a^{-(sl-ol)}}{m - ol} \tag{3.4}$$

where a, $m \in \mathbb{R}$, $a > 1.0$, $m > \max\{ol \mid ol \in OL\}$, where OL is the set consisting of each possible ol in the system. Here a^{ol} represents the estimated value of an object and a^{sl} represents the trustworthiness of a subject. TI approaches infinity (temptation becomes very large and difficult to resist) as ol approaches m.

To relate P_1 to TI, a sigmoid function:

$$P_1 = \frac{1}{1 + \exp((-k) \times (TI - n))} \tag{3.5}$$

is derived. Here n is the value of TI that makes $P_1 = 0.5$; k controls the slope of the P_1 curve with regard to TI. This function is chosen because of its similarity with the step function in the Bell-LaPadula model.

P_2 corresponds to the difference between the probability of unintentional disclosure and the probability of disclosure which the organisation is willing to accept for an access of a subject to an object. In the case that an object being accessed belongs to multiple categories, a simplified assumption is made such that the object is a monolithic entity, in which the differences among all the categories are computed and the maximum difference is used as P_2. Instead of a binary value, a fuzzy membership value is assigned to each subject to represent the "need-to-know" level of a subject for the object in each category.

The willingness index, WI can be computed using (3.4) with the fuzzy memberships in place of the sensitivity levels. Then the willingness of acceptance for an organisation when a subject accesses an object, W_c can be computed using (3.5) with WI in place of TI. Let P_{id_c} be the probability of unintentional disclosure for category c. Then, P_2 is defined as follows:

$$P_2 = \max\{P_{id_c}(1 - W_c) \mid c \text{ is a category}\} \tag{3.6}$$

Finally, the risk of a read access is computed using (3.1).

Review

The Fuzzy MLS model draws inspiration from fuzzy set theory to estimate and quantify risk in the traditional Bell-LaPadula model. This model continues to employ the risk based rationale, but it changes the access control model from a risk avoidance system to a risk management system by estimating the risk of each access with a risk scale.

This risk scale allows the model to adapt to environmental changes in several ways. Firstly, the boundaries between regions can be dynamically adjusted to control the system's global risk tolerance. Secondly, accesses that would have been denied under the traditional model may now be allowed if the associated mitigation measures can be provided. Thirdly, the risk function that maps each access to a point on the risk scale can be updated periodically to reflect the changes in the operational environment.

In practice, the security labels of the objects in a system are often set with a tendency towards higher secrecy to minimise the operational risk. Consequently, information sharing becomes more difficult. This over-classification problem can be addressed by making the label take uncertainty into account [2]. This is not possible in the traditional Bell-LaPadula model.

The Fuzzy MLS model however has some shortcomings; it only uses the security labels to estimate risk. As outlined in [8], there are many other factors that can affect the risk of an access, e.g., the access method (softcopy vs. hardcopy), the operational environment and the risk mitigation measures employed. The best way to identify, quantify and aggregate these factors remains an open research topic. Risk aggregation can be tricky as the sensitivity of a piece of information provided to a subject does not solely depend on itself, but also on other information that the subject can access [2]. Additionally, accesses to multiple pieces of low sensitivity information may allow a subject to infer information that is far more sensitive.

3.4.2 Economics based models

The idea of managing risk in access control using economic concepts is proposed in [8]. Each user in an organisation is allocated a number of risk tokens, which can be spent for operational needs. The number of risk tokens a user will get in the future depends on the return of investment of the user. Essentially, risk is viewed as a type of limited resource in an organisation and the access control problem is transformed into a resource allocation problem. Two economics based models are introduced: one based on the command economic system and the other based on the market economic system.

Definition of risk, damage and harm

New definitions for risk and two new terms, “damage” and “harm” are introduced in [8]. Risk is defined as the unnormalised probability of an access which can cause the loss of a secret, damage is defined as the cost of the loss, and harm is defined as the expected cost of the loss. These terms are related as follows:

$$\text{harm} = \text{damage} \times \text{risk} \tag{3.7}$$

The definition of risk is different from the one in the Fuzzy MLS model (3.1), which is restated as follows for ease of reference:

$$\text{risk} = (\text{value of damage, } V) \times (\text{probability of incurring the damage, } P)$$

A careful examination of both equations allows us to establish a mapping between the terms: the term “harm” here corresponds to the term “risk” in the Fuzzy MLS model; the term “damage” holds the same meaning in both cases; and the term “risk” here corresponds to the term “probability of incurring a damage” in the Fuzzy MLS model. In this thesis, the terms are used in the sense of the original report for ease of reference.

Three guiding principles

The author suggests that the new information protection systems should be risk based and outlines three principles that should be followed in building them. These principles are as follows:

1. Measure risk — The amount of risk associated with each access should be measured or at least estimated.

2. Mark an acceptable risk level — Risk avoidance (setting the acceptable risk level to zero) effectively stops all the information accesses because every access has an inherent amount of risk associated with it. The acceptable risk level should be set to a value that optimises the long-term benefit.

3. Maximise the information flow up to the acceptable level — In contrast to current systems which attempt to minimise the total risk, information in the

system should flow to the greatest extent compatible with the acceptable risk level in order to optimise gain.

Risk model

A risk model is required to estimate the risk of each access. The following factors should be considered in building this model:

Individual factors — User roles, security clearances, previous positions, etc.

Situational factors — Operational environment, access time, etc.

Technical factors — Hardware/software security measures, etc.

Types of accesses — Access method (softcopy vs. hardcopy), access duration, whether access is auditable, whether information can be redistributed, etc.

Temporal effects of consequences — Leaking the information on the budget allocated for a small project may result in a short-term risk but leaking the information on a national secret weapon may result in a long-term risk.

These factors have to be combined in a mathematical way to yield a formula that can be used to assign a risk value to each information access.

Model based on command economic system

In this model, the value of a token is pegged to risk [8]. For example,

> 1 token is pegged to the risk associated with softcopy access for a day
> to a document by an individual who is cleared to the secret level.

The value of the token is associated with the estimated risk incurred in the access, including the type and duration of the access and the security clearance of the individual. Yet, it does not consider the value of the document (the damage factor).

Using the risk model presented in Section 3.4.2, each access can be associated with a certain number of risk tokens. Higher risk accesses cost more. For example,

- Softcopy access for a day to a document by an individual who is cleared to top secret level costs 0.2 token.
- Hardcopy access to a document with a restriction on further distribution by an individual who is cleared to secret level costs 50 tokens.

When a piece of information is being produced, the producer will create the number of tokens that is commensurate with the acceptable risk level of the piece of information. For example, the production of a document that may be viewed in softcopy for 500 times by an individual who is cleared to secret level is always accompanied by the creation of 500 tokens. If the 500 tokens are spent by an individual who is cleared to secret level to print the document in hardcopy format for 10 times with a restriction of no further distribution, the acceptable risk level of the document would still be considered reached.

Additionally, the use of different types of tokens is necessary for different types of information. This is because all tokens are pegged using the same baseline. More sensitive information has less tolerable risk level and therefore has fewer tokens created with it.

To increase the liquidity of the market, a secondary market can be introduced to allow token exchange. This does not require any change of the model because the tolerable risk of the information has been controlled by the number of tokens created with it and other risk factors have been taken care of in the cost associated to each information access.

To distribute the risk tokens, the information producers create and distribute the risk tokens to different organisations based on their needs. Within an organisation, risk tokens are pushed down through the management hierarchy. Periodically, the distribution is reviewed based on the return on investment function. Other metrics can also be used to adjust the distribution of the risk tokens. An example of such metric is the utilisation function that measures the fraction of tokens that have been spent to purchase information accesses. This function can also serve as a measure on the merits of the information producers. If the utilisation fraction is near 100%, it means the organisation is almost reaping the full benefit of the information produced and vice versa.

Model based on market economic system

The main aspect that differentiates this model from the model based on the command economic system is the collapse of information specific tokens into two general types. This removes the controls that the producers have in setting the tolerable risk associated with the information (via manipulation of the number of tokens created). The reason for having more than one type of token is that different types of information require different protection profiles over time. This change requires the value of a risk token and the token distribution mechanism to be redefined.

In the model based on the command economic system, accessing to any information, regardless of its sensitivity level, would cost the same number of token for all individuals with the same clearance level. This is because the risk of an information access is only associated with the probability of causing unauthorised disclosure of the information. However, the amount of damage caused by unauthorised disclosure of information with different sensitivity levels are likely to be different. More sensitive information is likely to cause more damage. To control the amount of damage, the information producer can limit the number of tokens created along with an information. With the change from information-specific tokens to generic tokens, this is no longer possible.

To overcome this problem, the value of a token in this new model is changed to be associated with harm (damage$\times$risk) as defined in (3.7). To calculate harm, an additional damage model is required. For risk token creation and distribution, a new central authority that plays a similar role to the central bank in the real world is introduced. This central authority is also responsible for monitoring the health of the tokens by balancing demand and supply, and controlling the inflation and deflation rates.

Review

Although the author believes that the model based on the market economic system is superior to the model based on the command economic system, there are interesting features and tradeoffs in both models. As shown in Table 3.1, the value of a token is defined in different ways. The model based on the command

	Command economic system	Market economics system
Token value	pegged with risk	pegged with harm
Token type	information-specific	two generic types: short-term and long-term
Token creator	information producer	central authority (central bank)
Model required	a risk model only	a risk model and a damage model
Information risk	managed by limiting the number of tokens created with information	managed by maintaining the equilibrium of the information market

Table 3.1: The differences between the model based on the command economic system and the model based on the market economic system.

economic system has been deliberately simplified to introduce the new concept and to make way for the model based on the market economic system.

To have a fair comparison, the value of a token in the model based on the command economic system is associated with the amount of damage. The only remaining difference now is the controls the producers have in setting the tolerable damage associated with information in the model based on the command economic system. The information producers can be dishonest in their claims of the sensitivity levels of information by manipulating the number of tokens created with them. This may result in market dislocation. The model based on the market economic system is proposed to alleviate this problem by the introduction of a central bank, thus making the credibility of the central bank a key factor for this new model to operate properly. In other words, the author assumes that the risk of the central bank in being compromised is smaller than the risk of market dislocation caused by the dishonest behaviours of the information producers. This assumption may not hold true in certain operational environments, e.g., MANETs do not have a central trusted node and all nodes are exposed to security attacks.

Having said that, the three guiding principles in building a protection system as outlined in Section 3.4.2 are inspiring. Current protection systems always attempt to minimise the risk incurred in each access in the hope that the total risk of all accesses is below the acceptable risk threshold limit. These principles recognise that risk is inevitably incurred in each access and thus advocate managing the global risk. The acceptable risk threshold limit is first determined and information flow is encouraged all the way up to the acceptable limit to maximise the gain. This provides greater short-term flexibility in the access control. Information is made available to any user who is willing to pay the cost, yet the long-term behaviours of users remain under control as the token distribution is subject to the return on investment of each user.

Additionally, the distribution of risk tokens based on the return on investment encourages users to opt to access information in safer ways so as to reduce expenses. Thus, the levels of risk tolerance can be controlled in the number of tokens given to individuals; the fewer tokens, the more conservative the risk tolerance is. If the tokens are spent in the usage of a riskier (expensive) way to access information, individuals may not have sufficient tokens to accomplish other duties assigned to them.

There is a natural reluctance for human users to make decisions that have far reaching consequences, e.g., revoking the clearance of a user. The risk model helps by redefining the responsibilities of a human user in terms of security and operational duties. The security duty encompasses ensuring the integrity of the information entered to the risk model. This incremental information input to the risk model, which gradually changes the trust levels of a user, becomes less daunting in comparison to the immediate revocation of the clearance of a user. The operational duty is to ensure that the tokens are used or distributed wisely in the organisation/team, e.g., a manager may make an economic decision in distributing the tasks to his employees. However, the use of the risk model leads to an accountability issue: who is going to be responsible when something goes wrong? The risk model, the user or the higher level organisation? There is no obvious answer. Without accountability, the number of misuses is likely to increase.

In economics based models, the availability of information access is tightly linked to the risk tokens. A user who has spent all his budget is rendered useless until the next token distribution cycle. This can happen due to the use of an imperfect risk model, an imbalanced distribution of risk tokens or misbehaviours of users. In the former two cases, continual refinements have to be employed in a timely manner to avoid causing market dislocation. If it is due to the misbehaviours of users, simply denying the user's access request can be a logical answer but inappropriate in certain scenarios. For example, denying an information access to a front-line commander in a battlefield who has no budget left may result in fatalities.

Both models also assume that entities are rational and therefore will make the best decision among given choices. However, psychological and social research suggests otherwise; the human decision-making process is often suboptimal and irrational. For example, the "heuristics and biases" programme was started to investigate the idea of whether the decision-making process under uncertainty often rests on a limited number of simplified heuristics[1] or a complicated algorithm [59]. One of the results was the framing effect, which showed that the way a problem is formulated can have influence on the decision-making process [60]. A problem can emphasize a gain (30% of the people will pass the examination) or a loss (70% of the people will fail the examination). The former case generally leads to risk aversion behaviour while the latter case generally leads to risk seeking behaviour. Relationships among people can also affect the decision-making process. For example, a captain may choose not to report the misuse of authority among his soldiers to protect his soldiers from punishment or to preserve the reputation of his team. Other factors such as emotion, bias, mistake or incapability in judgement [61] can also creep into the decision-making process and cause chaos in the system.

Additionally, the report does not present any implementation example of core components, e.g., the models to calculate risk and damage. These models are inherently complicated to design and build. The contributing factors are difficult to measure, quantify, calculate and aggregate, e.g., how secure is an operating system? Should an access to a secret document be considered a short-term risk, a long-term risk or both? If both, what is the proportion of each risk? Even if building such models is possible, the task of fine tuning these models in a timely manner can be challenging. Furthermore, other security prerequisites on the infrastructures required to implement the system, as discussed in [8], ranging

[1]Heuristics are the simple yet efficient informal rules that humans use to make decisions.

from network protocol to technologies on tamper proof hardware, are difficult to engineer. The security of a system is only as strong as its weakest component.

In summary, the use of economic concepts may provide greater flexibility to protect information systems. However, there are doubts arising from its implementation and also with regard to some practical issues of the system. Indeed, the author also commented that "(they) fully expect that much of what (they) suggest can be proved unworkable, or no better than some different approach" and "(this) model may seem too extreme" [8]. Further research is required to further investigate this idea. Having said that, the 3M principles of building an information protection system: measure risk, mark the acceptable risk and maximise the information distribution to the acceptable level are inspiring thoughts.

3.4.3 Top-down hierarchical models

The models presented so far have access control policies which are specified in terms of the low-level corresponding enforcement mechanisms, e.g., protection bits, capabilities and access control lists. Each model implements a single specified policy but does not often capture all the protection requirements of a system.

The requirements of the policy languages

In [62], Woo et al. proposed two ideas: a logic based language that allows access control policies to be specified independently from the implementation mechanisms; and two composition operators that can be used to combine multiple access control policies. They also outlined a list of requirements for a language to be suitable for specifying access control policy. The language should:

1. be declarative and semantically independent from the implementation mechanisms.

2. be efficiently computable, hence allowing efficient authorisation evaluation.

3. allow the intended security properties to be easily specified.

4. allow the ways to handle authorisation to be easily specified when policies are non-monotonic, inconsistent, incomplete (coverage) or combined together.

Although it has been found later in [63] that the language proposed does not impose sufficient constraints to ensure that the specified policy is Turing decidable and therefore may not be implementable, their work has pioneered the use of high-level languages to specify abstract policies independently from the implementation mechanisms. In [64, 65], the Authorisation Specification Language (ASL), which is based on the stratified first order logic, became the first complete and computable policy specification language. There exist also other policy specification languages in the literature, e.g., Security Policy Language (SPL) [66] and XACML [67]. Refer to [68] for detail.

Policy hierarchy

In network management research, policies have become increasingly popular as a means of managing distributed systems. Here the term "policies" carries a much broader meaning; policies are "rules governing the choices in behaviour of a system (in general)" [69]. Therefore, policies encompass not only security-related rules, but also general management rules. For example, obligation policies which have the form of event triggered condition-action rules can be used to define adaptive management actions, e.g., change in the quality of service provided, resource allocation and backup policy and software installation. This difference is not important in the discussion of this thesis.

To cope with the growth in size and complexity of large distributed systems, there is a trend towards automating many aspects of management across distributed components. The concepts of viewing policy as an object and of policy hierarchy are first proposed in [6] and refined further in [7, 70]. The concept of viewing a policy as an object is about decoupling the policy from the components that are responsible for enforcing it (the implementation mechanisms) and viewing the policy as an independent reusable component [6]. This enables the behaviour of the system to change by simply changing the rules in the policy.

The concept of policy hierarchy recognises that policies exist at different levels of abstraction. It suggests that high-level policies can be derived from business goals and form the basis of multiple low-level policies [6, 7]. The ultimate objective is to develop a mechanism that allows the specification of a high-level policy to be analysed and translated automatically to low-level policies which can then be executed by the system. The number of policy hierarchy levels may vary among different models, yet the intuition remains the same. The high-level policies are refined to form low-level policies.

As an example, the policy model proposed in the International Technology Alliance (ITA) project is shown in Figure 3.3 [3]. The model consists of four layers: specification layer, abstract layer, concrete layer and executable layer. The specification layer consists of authoring and analysis tools that support the specification of high-level security policies in constrained natural languages. These policies are then refined into abstract policies. At the next layer, various formal methods are used to check the correctness and consistency of these abstract policies. The concrete layer is then responsible for refining the analysed policies into concrete policies which are then upheld by different components to meet the policy goals. The executable layer transforms these concrete policies into executable policies and distributes them to the implementation devices that are responsible for enforcing the policies. This bottom layer is also responsible for reporting the status and device discovery information back up to the concrete policy layer.

Policy refinement and policy conflict analysis

Policy refinement is the process of transforming high-level security goals into low-level policies that can be enforced by a system [6]. The refinement process involves the analysis of policy conflict, policy coverage and the determination of resources required to implement the policies [6].

One widely accepted policy refinement approach is the goal refinement approach proposed in [71]. The goal refinement approach consists of the process of identifying, recognising and instantiating refinement patterns. Once a refinement pattern has been identified and analysed for completeness and conflict, any policy that matches this pattern is certain to be complete and correct. Whilst it

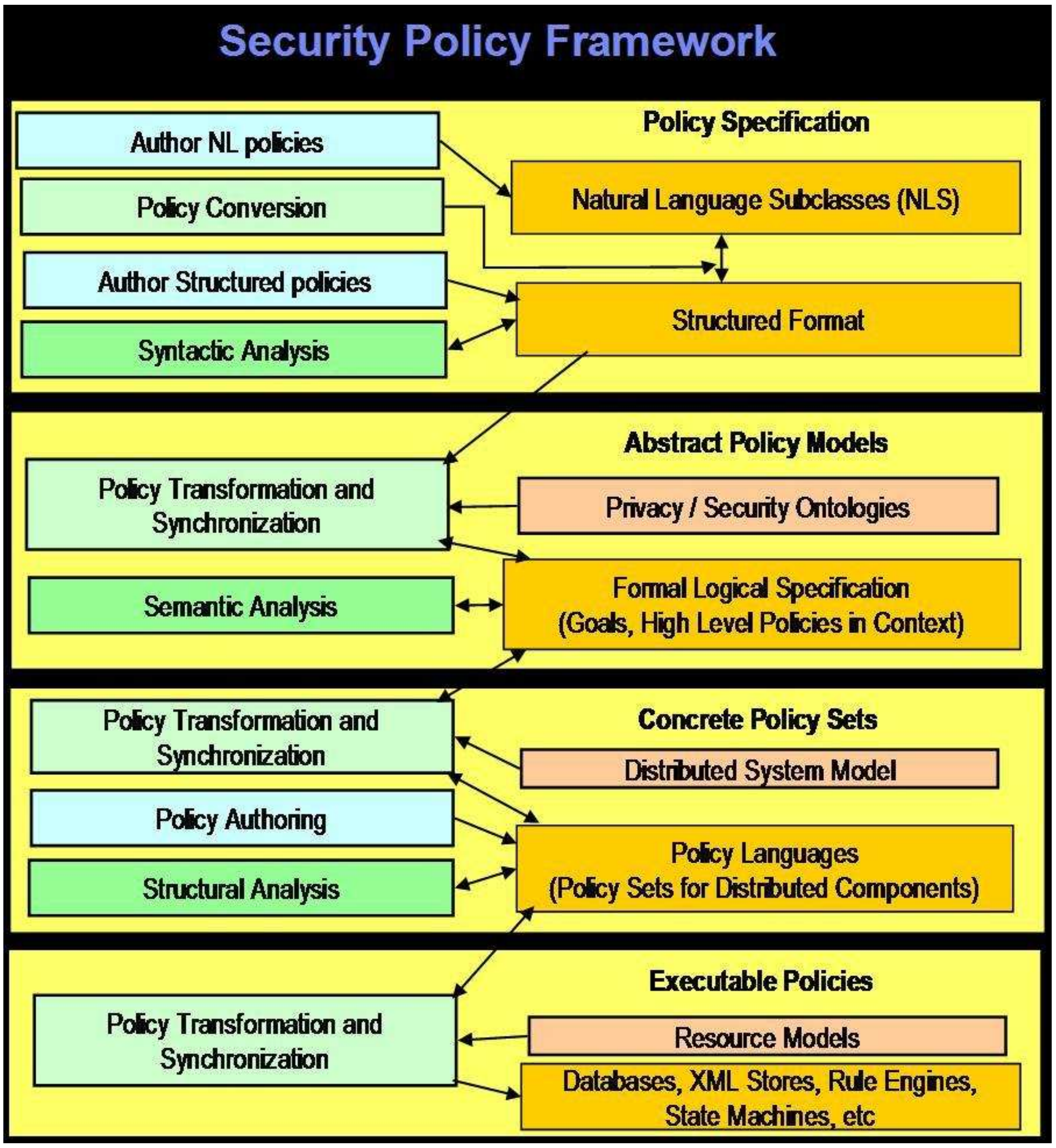

Security Policy Framework
Policy Specification
Author NL policies
Natural Language Subclasses (NLS)
Policy Conversion
Author Structured policies
Structured Format
Syntactic Analysis
Abstract Policy Models
Policy Transformation and Synchronization
Privacy / Security Ontologies
Semantic Analysis
Formal Logical Specification (Goals, High Level Policies in Context)
Concrete Policy Sets
Policy Transformation and Synchronization
Distributed System Model
Policy Authoring
Structural Analysis
Policy Languages (Policy Sets for Distributed Components)
Executable Policies
Policy Transformation and Synchronization
Resource Models
Databases, XML Stores, Rule Engines, State Machines, etc

Figure 3.3: The ITA policy model [3].

is desirable to have a fully automated refinement process, Moffett et al. argued that it is often infeasible to do this in many situations other than the most trivial scenarios [70].

Policy conflict analysis is the process of verifying whether the security policy is *consistent* and *complete* [69]. By consistent we mean that there is no conflict between the rules in the policies and with the capabilities of the underlying system. By complete we mean the policy implements all the high-level goals specified.

There are two categories of conflicts: modality conflicts and semantic conflicts. Modality conflicts arise when the rules in the security policies are inconsistent with one another. Modality conflicts can be divided into the following three categories based on the types of rules that are in conflict [72]:

1. Authorisation conflicts — conflicts that arise because both positive and negative authorisation rules exist for the same action, subject and object tuple. In other words, a subject is authorised (by the positive authorisation rule) as well as forbidden (by the negative authorisation rule) to perform the same action on an object.

2. Obligation conflicts — conflicts that arise because there is an obligation rule and a refrain (negative obligation) rule defined for an action that a subject is obligated to perform as well as refrained from performing on an object.

3. Unauthorised obligation conflicts — conflicts that arise when there are an obligation rule and a negative authorisation rule defined for an action that a subject is obligated but forbidden to perform on an object.

Having a positive authorisation rule and a refrain rule defined for the same action for a subject and an object is not considered as a conflict.

Poor policy specification is not the only cause of policy conflict. Organisation goals may be ambiguous and conflicting in nature, e.g., maximising resource utilisation vs. maximising resource availability. This will inevitably result in conflicts among policies that are derived from it.

To detect these conflicts, syntactic analysis can be applied to the policies to determine the overlap of subjects, targets and actions [72]. However, the existence of overlap only reveals the *potential* modality conflict because other constraints, such as time, might limit the applicability of the rules. Moreover, syntactic analysis is unable to detect application-specific conflicts, e.g., the separation of duty principle described in Section 3.1.4. To detect these conflicts, the conditions that may cause the conflicts are required to be specified as additional constraints on the policies. The occurrence of the conflicts may also depend on the state of the system. Analysing all these states to check for possible conflicts is often infeasible and therefore run-time analysis is still necessary.

Once these conflicts are detected, it is necessary to resolve them. Jajodia et al. suggested a few ways to handle the conflicts in [64]. The simplest way is to do nothing but flag an error condition. A better solution is to allow the positive authorisation policy to override the negative authorisation policy or vice versa. Often, the priority is given to the negative authorisation policy based on the assumption that preventing actions would incur less risk. Obviously this is not always true. For example, a positive authorisation policy can be an exception to a more general negative authorisation policy.

To alleviate this issue, priorities can be assigned to different policies explicitly [72]. When conflicts arise among policies, the highest priority policy is enforced. However, the task of priority assignment in itself is difficult. There is also a problem of breaking a tie if there are two or more policies with the same level of priority. This problem is exacerbated when there are multiple parties involved in defining and assigning policies. Inconsistency can easily arise as each party can have different preferences. An alternative approach is to define priority based on the specificity of the policy [73]. At the other extreme, meta policies are also being proposed as a way to define the precedence relationship among policies [72]. Whilst these resolution mechanisms provide more flexibility, they also make the task of ensuring policy consistency more complicated. There is still no known general mechanism that is able to detect and resolve conflicts among policies when arbitrary conditions are allowed.

3.5 Summary

Recent research [8] suggests that current static security policy models are not appropriate for many modern systems, especially when the operational environment is highly dynamic. Some models that provide more flexibility have been proposed [2, 8, 29]. These models are different from the static ones in two important aspects. Firstly, the risk-benefit tradeoff assessment on an information access request is not encoded in the security policy itself. Instead, an explicit risk model is used in these models to dynamically estimate the risk of an information access request to make better informed decisions. Secondly, the new model attempts to manage the total risk of a system as a whole, as opposed to the risk of each access individually in the traditional models. Users are allocated with an initial budget of risk tokens, which they may use on their discretion to access different information. The budget distribution is reviewed periodically based on the benefit gained from information access of each user. However, the models proposed are rather abstract. Many aspects of the models require further investigation. These include the way to allocate initial budget, the type of the market, the cost of the access, etc.

In the network management research, the top-down policy refinement approach has received much attention recently. The idea of this approach is to refine business goals to high-level security policies, which in turn are refined into low-level policies automatically. Whilst this conceptual idea is useful, the current policy conflict resolution mechanisms are still rather primitive. There is no general mechanism that is able to detect and resolve conflicts among policies when arbitrary conditions are allowed. As modern systems become more distributed, the distribution of the analysis procedures across the system can be problematic. The system may have a complex structure, with subsystems owned by different domains. This makes the policy refinement process much more complicated. In MANETs, subsystems can join, leave and rejoin at any time. This dynamic behaviour can easily cause conflicts among the policies.

The way we choose to approach the problem is a radical one. Here we investigate how a specific set of decisions may be generalised into an applicable security policy using Evolutionary Algorithms (EAs). A developed policy inference system could be doubly useful. The generated policy rules can be used on their own or used to verify the correctness of existing policies. We also explore the potential of EAs in dynamically updating security policies with the use of new decision examples. This feature will be essential in a highly dynamic operational environment like MANETs, where the risk factors are constantly changing.

Additionally, we observe that the risk-budget based policy models reviewed in Section 3.4 are really families of policies. Each instance in a policy family constrains the system and therefore affects the operational behaviour and effectiveness of the system in its own way. We introduce the notion of mission-specific policy and demonstrate how EAs can be used to search for the (near) optimal policies that fit a specific set of missions using simulation runs (instead of a set of decision examples).

3.6 Conclusions

This chapter summarises various influential security policies and models in the literature. It then introduces the top-down hierarchical policy model in which allows policies to be specified using high-level languages and then refined into low-level policies. Various issues related to the policy refinement and conflict analysis process are discussed. Lastly, it reviews the current state of the art in security policy development and reiterates the research objectives of the thesis.

Chapter 4

Learning Techniques

This chapter details the learning techniques used in this thesis. It begins with a brief introduction to evolutionary algorithms (EAs) and discusses two implementations of EAs, namely Genetic Programming (GP) [74] and Grammatical Evolution (GE) [75]. It then discusses how EAs can be extended to solve multi-objective optimisation (MOO) problems. The last section introduces fuzzy expert systems. These are used as examples in Section 5.2.4 to show how other learning techniques can be used in conjunction with EAs to improve the learning performance.

4.1 Evolutionary algorithms (EAs)

Evolutionary algorithms (EAs) are a set of heuristic search algorithms inspired by natural selection[1]. An initial population of individuals, which represents candidate solutions to the problem in question, is generated. This is typically done in a random fashion or, even better, seeded with some known good solutions to provide better search guidance. The individuals in the population are then repeatedly subjected to the evolutionary process that consists of the following steps:

[1]Loosely speaking natural selection states that individuals that are best adapted to the environment have better chances to survive and reproduce for the next generation [76].

1. Evaluation — Each individual is associated with a fitness value that measures how well it solves the problem.

2. Selection — Individuals are selected for reproduction according to their fitnesses (natural selection). This implements the notion of "survival of the fittest" [77].

3. Reproduction — Selected individuals are used to breed the population of the next generation using evolutionary operators.

This evolutionary process produces populations of individuals (candidate solutions) that are increasingly better suited to the environment (problem). Commonly, the stopping criterion is either that the maximum number of generations has been reached or a "good enough" solution has been found.

4.1.1 Evaluation

Each individual is associated with a fitness value that measures how well that individual solves the problem. In many applications this is formed by the application of some defined function to the candidate solution. In others, the "real world" acts as the cost function. For example, it is far easier to see how much power a program consumes by running the program and measuring its power consumption than by deriving and using a predictive model of power consumption (provided one has the electronics skills to carry out such measurements). The fitness value assigned to each individual provides a bias that is used to guide the search algorithm. Without this bias, the search is no better than random search [78]. From a practical perspective, the computational complexity of the fitness evaluation must be limited.

4.1.2 Selection

Individuals are selected based on their fitness values from the current population to breed individuals of the next generations; fitter individuals are selected preferentially over weaker ones. There are many selection techniques, the most commonly used of which are:

Roulette wheel selection technique [79] — This technique is also known as fitness proportional selection. The probability of an individual being selected is proportional to its fitness value. This technique is simple but the selection pressure is highly sensitive to the scaling effect of the fitness values. Typically, the variance of fitness values in the population at the beginning of a run is very high; there is only a small number of individuals that are much fitter than others. This technique heavily selects these individuals. Consequently, the diversity of the population decreases at a very high rate, often resulting in premature convergence. Towards the end of a run, the variance of fitness values in the population becomes very small as the individuals are very similar to one another. Thus, the selection becomes similar to random selection.

Rank based selection technique [80] — This technique uses fitness ranks instead of fitness values to determine the probability of an individual being selected. A typical implementation would assign probabilities according to an inverse linear relationship with rank. Thus, the probability of selecting the k-th fittest individual in a population of size n is given by $\frac{(n+1-k)}{N}$, where $N = \frac{n(n+1)}{2}$. To determine the fitness rank, however, requires sorting on the fitness values, which can be expensive if the population size is large.

Tournament selection technique [81] — This technique selects individuals by holding multiple tournaments. In each tournament, n individuals from the population are considered and the winner of the tournament (the one with the best fitness value) is selected. n is a tunable parameter known as the tournament size. The selection pressure increases with the tournament size. Binary tournament selection is the name given to this technique when $n = 2$.

Elitist selection technique [82, 83] — The selection method is not necessarily probabilistic. Elitist selection is an example of deterministic selection in which n of the fittest individuals in the current population are selected. Often, this technique is incorporated with other techniques so that a small portion of top performing individuals are guaranteed to be selected and copied to the population of the next generation. Other good performing individuals are selected probabilistically.

4.1.3 Reproduction

Evolutionary operators are applied on these selected individuals (commonly known as parents) probabilistically to produce individuals of the next generation (commonly known as children). Some common evolutionary operators used are:

Crossover — Elements within two or more selected individuals are exchanged to form new individuals.

Mutation — Elements within the selected individual are perturbed in some way. This serves to diversify solution elements in the population. Given an initial population, repeated applications of selection and crossover alone might otherwise not be able to reach parts of the search space.

Clone (Reproduce) — The selected individual is passed on to the next generation unchanged. This serves to preserve the best performing individuals (since specific instances of applying evolutionary operators do not reliably produce better individuals).

4.1.4 Implementations of EAs

Traditionally, EAs can be divided into the following four categories based on the individual representation adopted [84]:

1. Genetic Algorithm that uses fixed length binary string individuals.
2. Genetic Programming that uses tree structured individuals.
3. Evolution Strategies that use real valued vector individuals.
4. Evolutionary Programming that uses finite state machine individuals.

However, the boundaries between these categories become increasingly blurred. Often the properties from different categories are combined to suit the problem in question. For example, Binary Genetic Programming (BGP) [85] uses binary string individuals that encode tree structures and Cartesian Genetic Programming (CGP) [86] uses binary string individuals that encode graphs.

4.2 Genetic Programming (GP)

Genetic Programming (GP) is a form of EA in which an individual is typically a program represented by a tree structure. An example that implements the formula $(X \times Y) + (4 - (Y + 1))$ is shown in Figure 4.1.

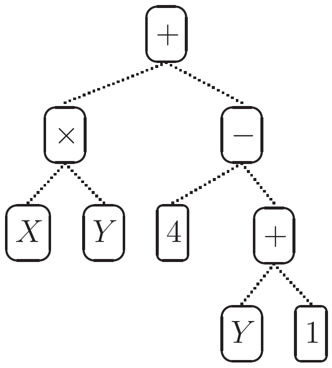


Figure 4.1: A GP individual that represents a formula $(X \times Y) + (4 - (Y + 1))$.

The tree nodes can be classified into two groups: the terminal set T and the function set F. The terminal set T consists of constants and variables (the leaf nodes) and the function set F consists of functions, operators and statements (the non-leaf nodes). An example of terminal set T and function set F which is sufficient to allow a description of the tree in Figure 4.1 is:

$$T = \{X, Y\} \cup \{1, 2 \ldots 10\}$$
$$F = \{+, -, \times, \div, \min, \max\}$$

These sets must fulfil the sufficiency and closure properties. The sufficiency property requires that the target solution for the problem in question can be represented with the elements in the sets. The closure property requires that the elements in the function set F can accept any value they may receive as input, including all the elements in terminal set T and any return values from other functions or operators. Practically, it is important to keep both sets small to prevent the search space from becoming too large.

In GP, the crossover operator is performed on two trees. A subtree in each tree is chosen randomly and swapped with the other. An example of crossover operation is depicted in Figure 4.2a, where the marked subtrees in the two trees are swapped with each other, resulting in two new trees. The mutation operator is performed on one tree. A node in a tree is randomly chosen and replaced with a randomly generated new node or subtree. An example of mutation operation is depicted in Figure 4.2b, where the marked node is replaced with a new subtree. Other operators also exist. For example, the reproduction operator copies the selected individual to the population of the next generation without changes and the permutation operator changes the node values of a subtree randomly.

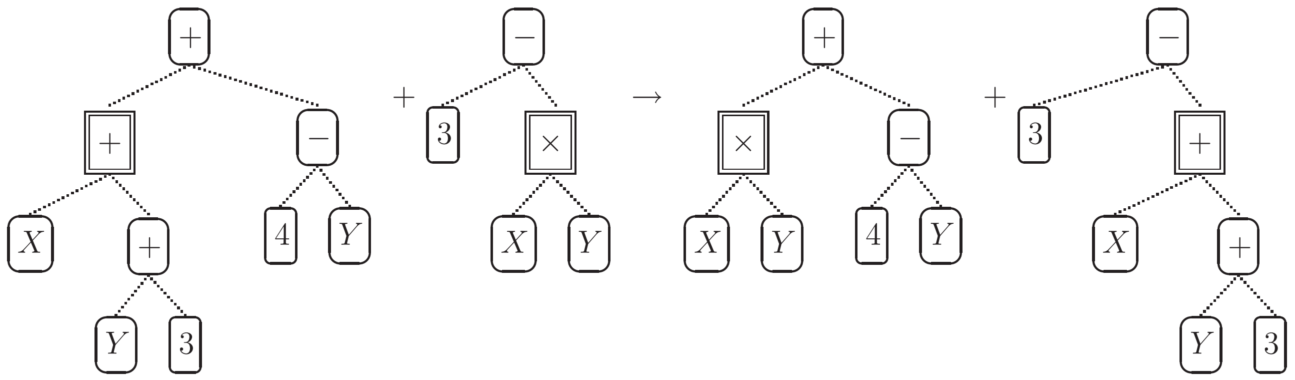
4.2.1 Extensions on GP

In the standard form of GP, there is no way to ensure that all the functions in each tree have inputs of the appropriate types (children). Strongly Typed Genetic Programming (STGP) is introduced in [87]. STGP augments each node with a type. A set of type rules is defined to specify how nodes in a tree can be connected with one another. For example, we may require that the "$\geq$" node to take two floating point inputs and return a Boolean value. The population initialisation and evolutionary operations must obey the type rules and only well typed individuals are maintained in the population.

Another way to ensure type conformance is to use a grammar to control the individual tree structure. However, the run-time grammar conformance checking is very expensive. In Section 4.3, Grammatical Evolution (GE), a technique that is able to evolve individuals in a grammar compliant way, is presented.

Automatically Defined Functions (ADFs) [88] are "building blocks" observed during the evolution process. They can be recognised and subsequently made available as units in the evolutionary process, effectively implementing an on-the-fly modularisation process.

Alternative individual representations have also been proposed. For example, Cartesian Genetic Programming (CGP) [86] uses integer strings as individuals


(a) Crossover operation.

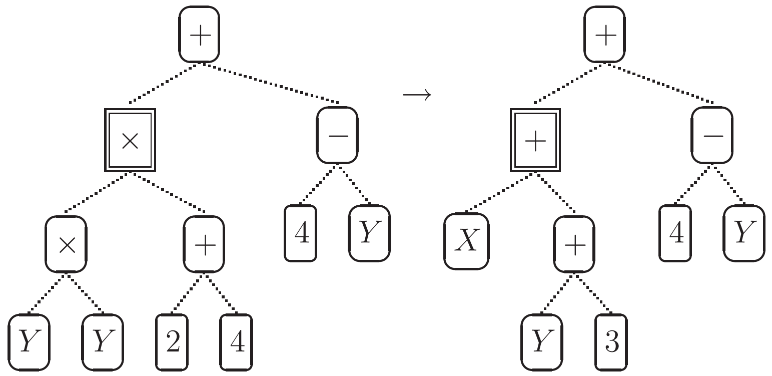

(b) Mutation operation.

Figure 4.2: The crossover and mutation operations in GP.

with each encoding a graph representation of a program. Constraints are placed on how the integer strings can evolve in order to ensure the validity of the underlying artifacts. This concept is very similar to the use of integer strings, which serve as indices to select which grammar rules are to be expanded in GE. Linear Genetic Programming [78, 89] uses individuals, which have variable length sequences of simple instructions that operate on one or two indexed variables (registers) or constants from a predefined set. PushGP [90], on the other hand, is proposed to evolve stack based execution instructions.

4.3 Grammatical Evolution (GE)

Grammatical Evolution (GE) uses variable length binary string individuals which serve as indices that map a set of Backus Naur Form (BNF) language grammar rules to programs [75]. These programs can be executed for fitness evaluation purposes. To solve a problem using GE, the language grammar must first be specified in BNF. The grammar of a language in BNF is represented as a tuple $\{N, T, S, P\}$ where N is the non-terminal set, T is the terminal set, S is a special element of N called start symbol which is expanded first, P is the set of production rules (also called derivation rules) that is used to guide the expansion from N to T. A production rule has the following format:

```
<non-terminal> ::= <expr-of-symbols-1>
                |  <expr-of-symbols-2>
                |  ...
```

The production rule states how the non-terminal symbol on the left hand side of the rule is to be substituted by one of the symbolic expressions on the right hand side of the rule. The "|" symbol is used to indicate the choice of expressions. An example of BNF grammar is as follows:

$$N = \{\texttt{<expr>}, \texttt{<op>}, \texttt{<var>}\}$$
$$T = \{+, -, \times, \div, X, Y, (,)\}$$
$$S = \texttt{<expr>}$$

and P consists of a set of production rules as follows:

```
<expr> ::= <expr> <op> <expr>         (0)
        |  (<expr> <op> <expr>)       (1)
        |  <var>                      (2)

<op>   ::= +                          (0)
        |  −                          (1)
        |  ×                          (2)
        |  ÷                          (3)

<var>  ::= X                          (0)
        |  Y                          (1)
```

This grammar defines a set of arithmetic expressions over the variables X and Y.

4.3.1 Genotype-phenotype mapping

The mapping process between the variable length binary string individuals and programs using BNF grammar is known as genotype-phenotype mapping[1]. This is best illustrated with an example. In each individual (genome), every eight consecutive bits are viewed as an integer which is more commonly known as a codon. Consider an individual with the following codons:

$$\{10, 5, 51, 8, 16, 49, 30, 18\}$$

As mentioned, the mapping process begins with the start symbol. The codon value is read to determine the usage of a certain production rule to expand the symbol using a modulus mapping function as follows:

$$\text{rule} = (\text{codon value}) \bmod (\text{total number of applicable production rules})$$

[1]In the biological world, a genotype is the internally coded information (genes) that is inherent in an individual. The phenotype is the "observable characteristics" of an individual that can be influenced by the genotype and environment. For example, the genotype of human being is the DNA and an example of the phenotype is his eye colour.

Assuming the start symbol is `<expr>`, the first codon value is 10 and there are three immediately applicable production rules for `<expr>`, therefore 10 mod 3 = 1. The production rule 1 is used to replace `<expr>` with (`<expr> <op> <expr>`). Assuming leftmost derivation is used, the leftmost `<expr>` is first expanded as "(" is a terminal. The second codon value is used to select the production rules. As 5 mod 3 = 2, the production rule 2 is selected and the leftmost `<expr>` is replaced by `<var>` resulting in (`<var> <op> <expr>`). The next non-terminal to be expanded is `<var>`. As the next codon is 51 and the number of rule choices now available is 2, `<var>` is replaced with Y.

The mapping process continues until a complete program is generated, i.e., when all the non-terminals have been transformed into terminals. However, some problems may arise during this process. Firstly, there is a possibility that the codon values may run out before the entire mapping process is complete. To overcome this problem, an individual can be wrapped around to reuse the codon value, i.e., the reading of the last codon (18 in the example) will be followed by the first codon (10 in the example).

If the context free grammar used is recursive, the mapping process can also be indefinitely long. Consider the production rules for `<expr>`. If the codon value always maps onto the production rule 0 or 1, the mapping process is never ending. The `<expr>` is replaced by either `<expr> <op> <expr>` or (`<expr> <op> <expr>`). In both cases, the next leftmost non-terminal is again `<expr>` itself. This problem can be resolved by setting an upper bound on the number of times the production rules can be performed. In the original algorithm, this is achieved indirectly by setting a limit on the number of times the individual can be wrapped around and thus a limit on the number of codons available. If a complete program is not fully generated (the non-terminals have not been transformed to terminals) when the wrap around threshold is reached, this individual is given the lowest possible fitness value to decrease the probability of this individual being subsequently selected.

4.3.2 Extensions on GE

The greatest advantage of GE is that it allows artifacts to be evolved in a grammar compliant way and many solution spaces can be defined using a grammar. Mechanisms for explicit prevention or repair of invalid artifacts are therefore not required.

The use of a grammar also makes the languages and search algorithms become independent components. The program generation in arbitrary languages is possible by simply changing the BNF grammar. The search algorithm can be replaced with other algorithms. Instead of using Genetic Algorithm, it is possible to use other search techniques. For example, Grammatical Swarm [91] uses Particle Swarm Optimisation to carry out the search. This feature allows GE to reap the advantages of improvements in any evolutionary algorithm.

Although the lengths of individuals can be set within a predefined range, this does not mean that the sizes of the generated programs will be similar. In many problems, it has been found that many of the randomly generated individuals fail to complete the genotype-phenotype mapping in the initial population generation, even when the individuals are allowed to wrap around up to thirty times [92].

To overcome this problem, a sensible method [92] is proposed to take the grammar into account in the initialisation process. It is modelled after the popular *ramp-half-and-half* initialisation method [88], which is able to generate an initial population that consists of 50% full trees and the other 50% partial trees with various heights and shapes within a predefined height range. This is achieved by first assigning each rule in the grammar with two properties: the minimum number of mappings required for the non-terminal on the left hand side of the rule to be completely mapped to terminals and whether the rule is recursive. As each derivative tree is built, only rules that can fit the remaining height of the tree are selected. To generate a full tree, the recursive rules are always chosen whenever possible. To generate a partial tree, the recursive and non-recursive rules are chosen with equal probabilities. The final step in the initialisation is to reverse map the nodes of the tree to the corresponding codons.

Additionally, the traditional one-point crossover, which randomly chooses a point on each of the two selected individuals and swaps all data beyond that point, may not be very effective in GE. This is because the chosen crossover point may be at a position that is after the effective length of an individual (the portion of the individual that is actually used to select the rules) and render the crossover operation ineffective. To overcome this problem, effective crossover is introduced. This operation restricts the crossover points chosen to be within the effective length of the selected individuals.

4.4 Multi-objective evolutionary algorithms

In many practical problems, a desire to optimise more than one objective is common, some of which can be in conflict with others. Instead of having a single fitness score, each possible solution has a vector of fitness scores, one per objective. This is typically referred to as a multi-objective optimisation (MOO) problem. To visualise this, the concepts of the solution space S and objective space O are used. An example of S and O for a minimisation with two objectives is shown in Figure 4.3. Each point $s \in S$ represents a possible solution to the problem in question[1]. The fitness function f maps a point $s \in S$ to a point $o \in O$ such that the location of the point o represents how well s meets the objectives. The fitness function f is surjective (onto) but not injective (one-to-one); each point in S is mapped to one point in O and multiple points in S can be mapped to the same point in O. The region in which all the implementable solutions reside is known as the feasible solution region.

The aim of multi-objective evolutionary algorithms (MOEAs) is to discover the set of solutions with the optimal tradeoffs between all the objectives. Referring to the above example, these solutions are on the dotted lines in O in Figure 4.3.

4.4.1 Weighted sum of fitness functions

One widely used approach to solve MOO problems is to use a weighted sum of the individual fitness functions as the overall fitness function for the problem, as

[1]For ease of illustration, our solution space comprises individuals with two factor values. However, the solution and objective spaces need not have the same number of dimensions.

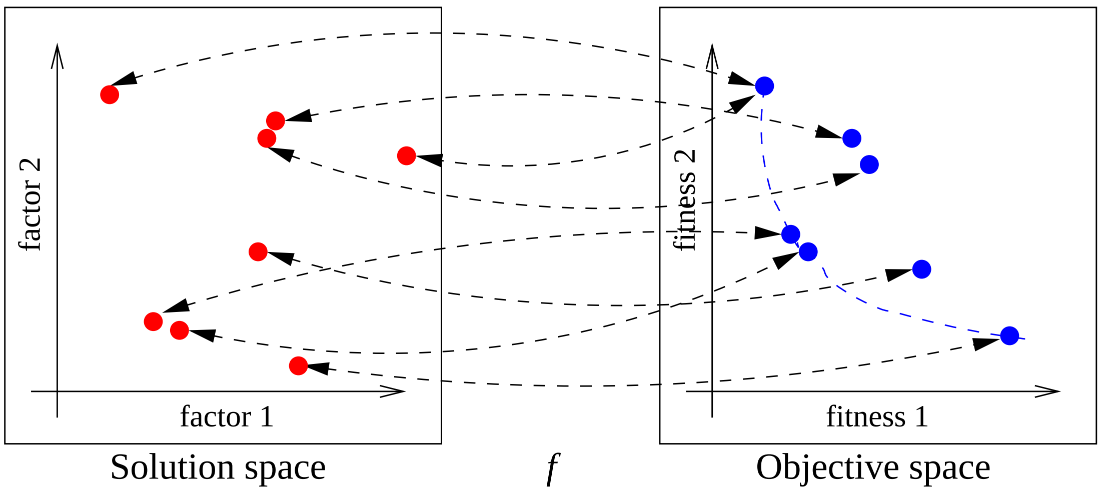


Figure 4.3: A mapping between solution space S to objective space O.

indicated below:

$$f(x) = \sum_{m=1}^{n} w_m f_m(x) \tag{4.1}$$

where w_m is a positive scalar weight for f_m. However, this approach requires the relative weights of all the objectives to be predefined. This is often difficult. For example, how many times higher/lower is the risk of an unauthorised disclosure of one information element in comparison to the risk of a soldier being killed? This is like comparing apples with oranges; there is inevitably some degree of subjectivity and arbitrariness in the weight assignments. A more principled and conceptually clearer approach would be advantageous.

Each chosen set of weights essentially defines a problem landscape via the single fitness function defined in (4.1). The space can then be searched using any appropriate technique. If the tradeoffs among objectives change, the technique has to be rerun to search for the new optimum solution. This can be very inefficient.

Additionally, for a certain set of problems, this approach is unable to discover all solutions with different levels of tradeoffs. Consider a two objective minimisation problem with its objective space shown in Figure 4.4. For a chosen set of weights (w_1, w_2), a line with gradient $-w_2/w_1$ can be drawn to represent the set of solutions that have the same fitness value. Geometrically, the process of minimising the fitness value can been seen as shifting this line towards the origin as shown in Figure 4.4. The optimal solution for any given set of weights would be the point that lies at the tangent to the feasible region. By changing the relative weights of the objectives, different solutions may be considered as the optimal. However, as there is no such line which can be defined such that the line passes through the points in the concave region, the set of solutions that lie at the concave part of the feasible region can never be discovered.

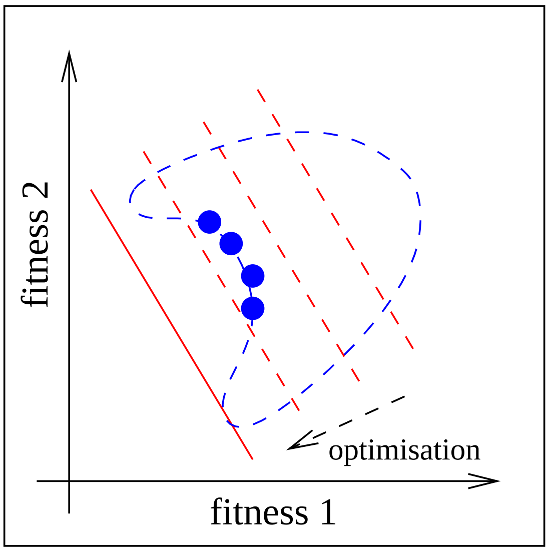


Figure 4.4: The set of solutions that lie at the concave part of the feasible region can never be discovered.

4.4.2 Pareto front based approaches

Pareto front based approaches search for multiple optimal solutions in one single evolutionary run using the Pareto dominance concept in the fitness evaluation of each solution (individual). A Pareto dominance, $\succ$ relation between two solutions, x and y is defined as follows:

Pareto dominance, $\succ$. *Let f_x and f_y be the fitness vectors of x and y, $>$ denote "is fitter than" relation, $x \succ y \Leftrightarrow (\forall i \cdot f_x[i] \geq f_y[i]) \wedge (\exists i \cdot f_x[i] > f_y[i])$ where $f[i]$ is the fitness score of the i-th objective.*

In other words, x dominates y if and only if x is better than y in at least one aspect and is at least equally good in all other aspects. This defines a partial order relation on the solution space. There always exists a subset of solutions that are not dominated by any other solutions. This subset represents the best solutions possible and is known as the *Pareto front* or the *Pareto optimal set*.

The aim of Pareto front based approaches is to converge the individuals in the population to the Pareto optimal set of solutions. Within the population of each generation, there is always a subset in which solutions are not dominated by any other solution in the population. The aim is to make this non-dominated subset a better approximation of the Pareto optimal set than the one in the previous generation. To have a good approximation to the Pareto optimal set, these approaches also attempt to maximise the diversity among the solutions in the non-dominated subset. Consider an optimisation problem with two objectives such as in Figure 4.5. Let the curved line represent all the solutions with the best possible achievable tradeoffs between the two objectives, i.e., the real Pareto front. Then, MOEAs attempt to converge the individuals (points) in the population to be as near to the Pareto front as possible and also as diversely spread on the Pareto front as possible.

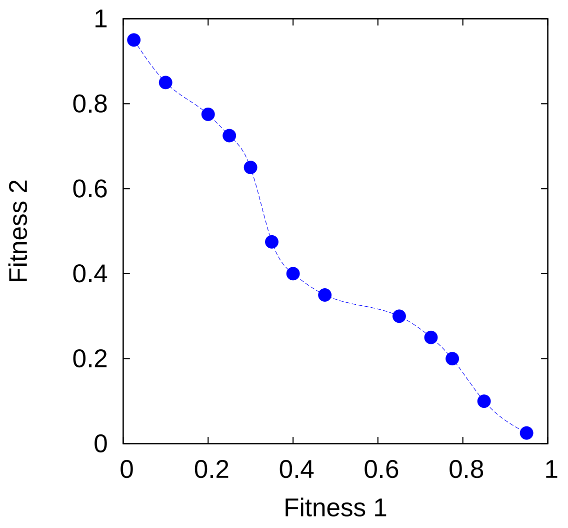


Figure 4.5: The objective of a Pareto front based approach is to approximate the Pareto front of the solutions with the individuals in the population.

4.4.3 Advantages of Pareto front based approaches

Pareto front based approaches have several advantages over the traditional weighted sum of fitness functions approach:

- The weights that define the tradeoffs among different objectives are no longer required to be determined a priori. Such a determination is difficult as it requires a deep understanding of the problem domain.

- The Pareto front can reveal the relationship between different objectives, which may be difficult to obtain otherwise. Such information is helpful in guiding a decision maker to choose the optimal solution for the problem from the Pareto optimal set.

- The set of Pareto optimal solutions can be saved and be retrieved later. Such a retrieval may be necessary if a change in circumstance requires a different set of tradeoffs and therefore a different solution.

- Optimising for multiple fitness scores tends to preserve the diversity of the population, which prevents the population from being trapped in a local optima and increases the chance of finding better solutions.

4.4.4 Implementations of Pareto front based approaches

There are many different implementations. Two of the most popular implementations, SPEA2 [93] and NSGA2 [94], are presented here. As EAs are stochastic in nature, there is no guarantee that the population will converge to the real Pareto front, i.e., the output is only an approximation of the real Pareto front.

Strength Pareto Evolutionary Algorithm 2 (SPEA2)

As suggested by its name, SPEA2 is an improvement over its previous implementation, SPEA [95]. SPEA2 is an elitist approach which uses a fixed size archive to maintain non-dominated individuals found in each generation. Binary tournament selection is then used to select individuals to produce offspring population using evolutionary operators. The best n individuals from the union of the archive and offspring population become the population of the next generation, where n is the population size.

As the archive size is fixed, three possible cases can arise: the number of non-dominated individuals is equal to, less than, or more than the archive size. In the first case, there is no additional work required.

If the number of non-dominated solutions is less than the archive size, the remaining vacancies are filled with non-dominated solutions. The selection is based on the fitness values of the individuals. The fitness value of an individual is determined by two factors:

1. Raw fitness — the sum of the strength of individuals it is dominated by. The strength of an individual is the number of individuals it dominates.

2. Density estimation — the inverse of the Euclidean distance of the individual to the *k-th* nearest neighbour in the objective space, where k is usually set to be the square root of the sum of population size and archive size.

In SPEA2, a smaller fitness value means fitter.

If the number of non-dominated solutions is more than the archive size, the non-dominated solution that has the shortest Euclidean distance to another individual in the objective space is dropped. If two solutions have the same distance to their nearest neighbours, the tie is broken by comparing their distances to their second nearest neighbours and so forth. This process is iterated until the non-dominated solutions can fit into the archive. Essentially, the goal is to fill the archive with non-dominated solutions as uniformly and widely distributed over the objective space as possible. Referring to Figure 4.6, an example of the formed Pareto front and the solutions that are to be kept in the archive is depicted.

Non-dominated Sorting Genetic Algorithm 2 (NSGA2)

Similarly to SPEA2, NSGA2 is an improvement algorithm over its previous implementation, NSGA [96]. NSGA2 is also an elitist algorithm that uses an archive to preserve the non-dominated solutions. The main distinction between these two algorithms is in the way they preserve the elites.

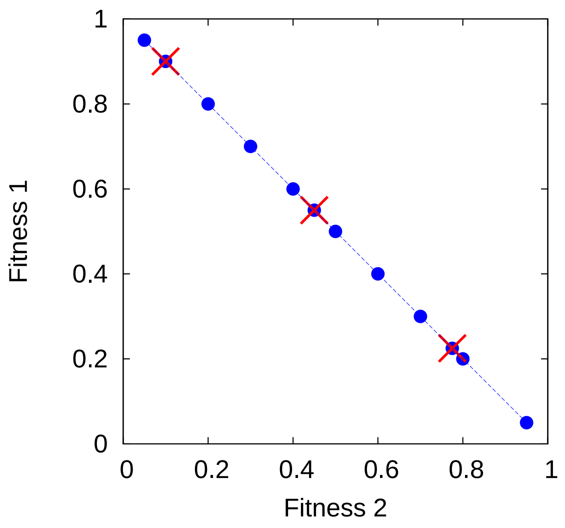


Figure 4.6: The archive truncation process in SPEA2 removes the non-dominated individuals that are closest to others *iteratively* until these individuals can fit into the archive. Assuming the archive size is 10, the truncation process removes the individuals that are crossed out.

In each generation, NSGA2 uses the Pareto dominance relation to divide the population into partitions with different rank levels. The non-dominated solutions in the population are first assigned to partition rank 1; the non-dominated solutions in the remaining population are assigned to partition rank 2, and so forth. This process is continued until all solutions are assigned to a partition. The rank of the partition is then used as the order to admit solutions into the archive; partitions with better (lower) rank are admitted first. Solutions are admitted on a partition by partition basis until the archive is filled or has exceeded its maximum capacity.

If the archive size has exceeded its maximum capacity, some of the solutions from the last admitted partition are removed based on the crowding distance. The crowding distance of a solution is the average distance of two adjacent solutions on either side of the solution along each of the objectives in the objective space. The individual with the smallest crowding distance is removed iteratively until the solutions can fit the archive.

Individuals are then selected from the archive using crowded binary tournament to produce an offspring population of the same size using evolutionary operators. The crowded binary tournament selects the better of two randomly selected individuals, where better means the individual with lower rank or the one with lower density measure (higher crowding distance) in cases where both of them have equal rank. The union of the archive and the offspring population forms the population of the next generation.

4.5 Relevant applications of GP and GE

This section reviews some of the applications of GP and GE in security related domains including rule inference system, intrusion and anomaly detection systems, security protocols and cryptography.

4.5.1 Rule inference system

Rule inferencing is the process of extracting a set of rules from a set of observations. In [97] a GP based rule inference system called LOGENPRO (LOGic grammar based GENetic PROgramming rule inference system) is proposed. In the experiment, the results show that LOGENPRO is able to outperform some Induction Logic Programming (ILP) systems in inferring rule sets. In [98] an experiment on coevolution between rules and fuzzy membership variables using GP is designed. The experimental results show that the output set of rules and variables are well adapted to one another. In [99] an attempt is made to invent a generic rule induction algorithm using grammar based GP. The results are found to be competitive with some well known manually designed rule induction algorithms.

GE is used to learn investment strategy (trading rules) using the market stock indices in [100–102]. The results show the investment strategy inferred yields profitable performance for the trading periods analysed. In [103, 104] GE is used to evolve rules for foreign exchanges. The rules learnt outperform the benchmark buy and hold strategy over all the trading periods analysed.

4.5.2 Intrusion and anomaly detection systems

EAs have also been used to discover useful patterns of system features that describe the behaviour of a system. These discovered features are then used to recognise anomalies and intrusions.

In [105] GP is used to evolve an intrusion detection program. In comparison with the results obtained using other machine learning techniques (more precisely, support vector machine (SVM) and decision tree techniques), GP is found to be able to generate a program that is more compact, efficient and selective in choosing the relevant input features. In [106] GP is shown to be able to evolve a set of autonomous agents that are used to implement an intrusion detection system. In [107] GE is also shown to be effective in generating an intrusion detection program.

Evolving an intrusion detection system or a computer program for MANETs using EAs is a relatively new domain. Often, MOEAs are used to search for programs that are not only optimal in accomplishing functional objectives, but are also optimal against other non-functional objectives, e.g., the program size, memory requirement and power consumption. These objectives are important considering that the operational environment may have very limited resources.

In [108] an MOEA based framework that is able to evolve efficient distributed programs for sensor networks is presented. The node election problem (selecting one node from a group of nodes in the network) with the aim of minimising the number of message transmissions is used to demonstrate the feasibility of the framework. In [109] GE is used to evolve an intrusion detection system for MANETs. The results show that the approach is promising.

4.5.3 Security protocols

Genetic Algorithm has been used to synthesise abstract security protocols in a small subset of the Burrows-Abadi-Needham (BAN) logic automatically [110]. This work has been extended to cover more complicated protocols in the BAN logic [111] as well as in the more sophisticated Syverson-van-Oorschot (SvO) logic [112]. The results show that Genetic Algorithm is able to synthesise abstract security protocols that are provably correct with respect to the logic used.

The use of binary string individuals which are treated as integer strings, the use of modulus operator to convert these integers to multiple indices within bounded ranges, and the use of these indices to select which element it represents in each protocol component are very similar to GE.

4.5.4 Cryptography

In [113] GP is applied to evolve a block cipher. In [114] an extremely lightweight and fast block cipher is successfully evolved using GP. This block cipher is competitive with the well respected Tiny Encryption Algorithm (TEA) [115] in terms of security strength and speed.

In [116] GE is shown to be able to evolve Boolean functions with cryptographic significance. The design of Boolean functions for cryptographic schemes is difficult because the desired properties of the functions are complicated and often conflicting with one another.

In [117] GP is shown to be able to evolve a circuit for the Quantum Fourier Transform, which is an important building block in Shor's algorithm [118, 119]. The discovery of this algorithm is significant as it is able to solve factorisation and discrete logarithms in polynomial time. Many widely respected encryption algorithms rely on these problems being computationally intractable.

4.6 Fuzzy expert systems

Unlike in the computer world where all the operations are essentially binary, our world is full of ambiguities. The concept of fuzzy logic was conceived by Zadeh as a way to deal with this ambiguity [120]. Instead of determining a statement as being either true or false, a degree of truth is associated with the statement using a real value in the range $[0, 1]$ where 0 means absolute falsehood and 1 means absolute truth.

4.6.1 Fuzzy set theory

In classical set theory, an element x can either be in a set S, or in the complement of the set S', but not both. Fuzzy set theory extends the classical set theory to allow partial membership. A value in the range $[0, 1]$ is introduced to describe the membership of x, with 0 representing complete non-membership, 1 representing absolute complete membership and values in between representing the degree of partial membership [121]. Often, this membership assignment is achieved using a function called the membership function.

Three basic fuzzy set operators, namely *union* ($\cup$), *intersection* ($\cap$) and *complement* ($'$) are defined. These are the analogues of the classical set operators. Let $\mu_A(x)$ and $\mu_B(x)$ be the fuzzy membership functions of the fuzzy sets A and B. Then, these operators are defined as follows [120]:

$$\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x)) \tag{4.2}$$

$$\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x)) \tag{4.3}$$

$$\mu_{A'}(x) = 1 - \mu_A(x) \tag{4.4}$$

There is a shortcoming with these definitions: the result of the operations depends only on one of the potentially many operands [122]. There are other definitions proposed for these operators. Many of these proposals attempt to redefine the *union* and *intersection* operators, but leave the *complement* operator unchanged.

In the same paper, two additional operators, namely *algebraic sum* ($\oplus$) and *algebraic product* ($\otimes$), are introduced and defined as:

$$\mu_{A\oplus B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x) \tag{4.5}$$

$$\mu_{A\otimes B}(x) = \mu_A(x)\mu_B(x) \tag{4.6}$$

As the $\oplus$ and $\otimes$ are equivalent operators to $\cup$ and $\cap$ in classical set theory respectively, these operators are sometimes viewed as alternative definitions for $\cup$ and $\cap$ operators. Indeed, these definitions are used in the Fuzzy MLS model introduced in Section 3.4.1. These operators are different in fuzzy set theory as $\mu(x)$ is no longer a binary value.

Yager proposed an alternative set of interpretations for $\cup$ and $\cap$ operators in [123] as follows:

$$\mu_{A\cup B}(x) = \min(1, (\mu_A(x)^w + \mu_B(x)^w)^{\frac{1}{w}}) \tag{4.7}$$

$$\mu_{A\cap B}(x) = 1 - \min(1, ((1-\mu_A(x))^w + (1-\mu_B(x))^w)^{\frac{1}{w}}) \tag{4.8}$$

with $w \geq 1$. A comparison of the behaviours of these interpretations is shown in the Figure 4.7 [122]. Refer to [124, 125] for other interpretations.

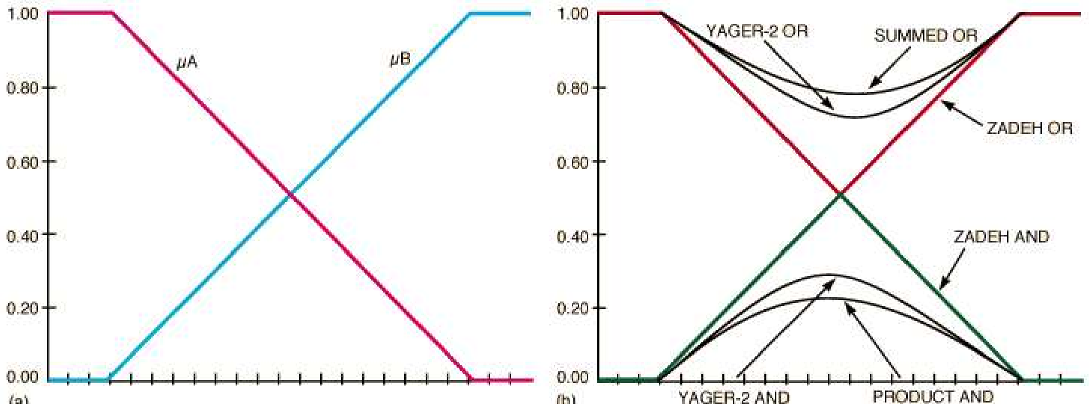


Figure 4.7: Various interpretations of the fuzzy operations.

In all these interpretations, the behaviours of the operators are the same as in the classical set theory if the membership value is restricted to only binary value (0 or 1). Therefore, fuzzy set theory and logic can be thought of as a generalisation of classical set theory and logic. Many of the interpretations proposed attempt to preserve the laws defined in classical set theory, e.g., commutative law, distributivity law and de Morgan's law. However, the law of contradiction $A \cup A' \equiv U$ (a set and its complement must establish the universe) and law of excluded middle, $A \cap A' \equiv \emptyset$ (an element can either be in a set or not, but not both) are not established in fuzzy set theory. Indeed, fuzzy set theory can exist only if these two laws do not hold.

Lastly, fuzzy set theory ought not to be confused with probability theory. While they are similar mainly because of the use of values in the range $[0, 1]$, there is no restriction on fuzzy set theory to enforce the concept that the sum of all memberships has to be equal to 1, as in probability theory.

4.6.2 Inference process

One major application of fuzzy logic is to build fuzzy expert systems. A fuzzy expert system consists of a set of fuzzy membership functions and inference rules that are used to reason about data [121]. These systems have achieved many successes in different fields, including automatic control systems, pattern recognition and data analysis. The inference rules take the following form:

`IF` X `is low and` Y `is medium THEN` Z `is high`

where X, Y are the input variables and Z is the output variable. The `low`, `medium` and `high` are the membership functions defined on X, Y and Z respectively. The antecedent (the part between `IF` and `THEN`) is the condition that describes the degree of truth and the degree of application of this rule. The conclusion (the part after `THEN`) assigns membership functions to the output variables.

Applying these inference rules to the data, we can infer the values of output variables for any given specific values of input variables. This inference process consists of the following steps [4, 121]:

1. Fuzzification — The input values are mapped to fuzzy membership values using the membership functions. In each rule, if the antecedent consists of more than one input variable, the input fuzzy membership values are aggregated using fuzzy set operators. The interpretation of these operators is a design issue for the system.

2. Inference — The aggregated fuzzy membership value is used to determine the degree of application of each rule. The fuzzy membership value is applied to each output variable for each rule. There are two commonly used inference methods: *min* and *product*, which are equivalent to Zadeh's *intersection* and *algebraic product* fuzzy operators respectively.

3. Composition — The fuzzy membership values assigned to the output variables in all the rules are combined to form a single fuzzy membership value. There are two commonly used composition methods, *max* and *sum*. The choice of composition method used often depends on which inference method is employed in the previous step. If the *min* inference method

is used, the *max* composition method, which is essentially equivalent to Zadeh's *union* operator, is typically used. If the *product* inference method is used, the *sum* composition method, which is essentially equivalent to Zadeh's *algebraic sum* operator, is typically used. An alternative interpretation of the *sum* composition method is the sum of the fuzzy membership values of the output variables in all the rules. This interpretation has the advantage of a simpler computation at the price of a lower accuracy.

4. Defuzzification — This is simply the reverse process of fuzzification. The fuzzy output membership value is converted into a discrete output value. There are many defuzzification methods; the common ones are *centroid*, *max* and *average of max* methods. In the *centroid* method, the output value is the centre of gravity of the fuzzy output membership value. In the *max* method, the output value is one of the variable values in which the fuzzy output membership value is maximum. In the *average of max* method, the output value is the average of all the variable values in which the fuzzy output membership value is maximum.

An example of the inference process is shown in Figure 4.8.

4.7 Conclusions

This chapter presents a survey on the learning techniques that are used in this thesis. It begins with a discussion on EAs with an emphasis on the two implementations: GP and GE. It then discusses how EAs can be extended to solve the multi-objective optimisation problems. The last section discusses fuzzy expert systems that are built on the ground of fuzzy inference rules. The fuzzy set concept is used as an example in Section 5.2.4 to show how other learning techniques can be used in conjunction with EAs to improve the learning performance.

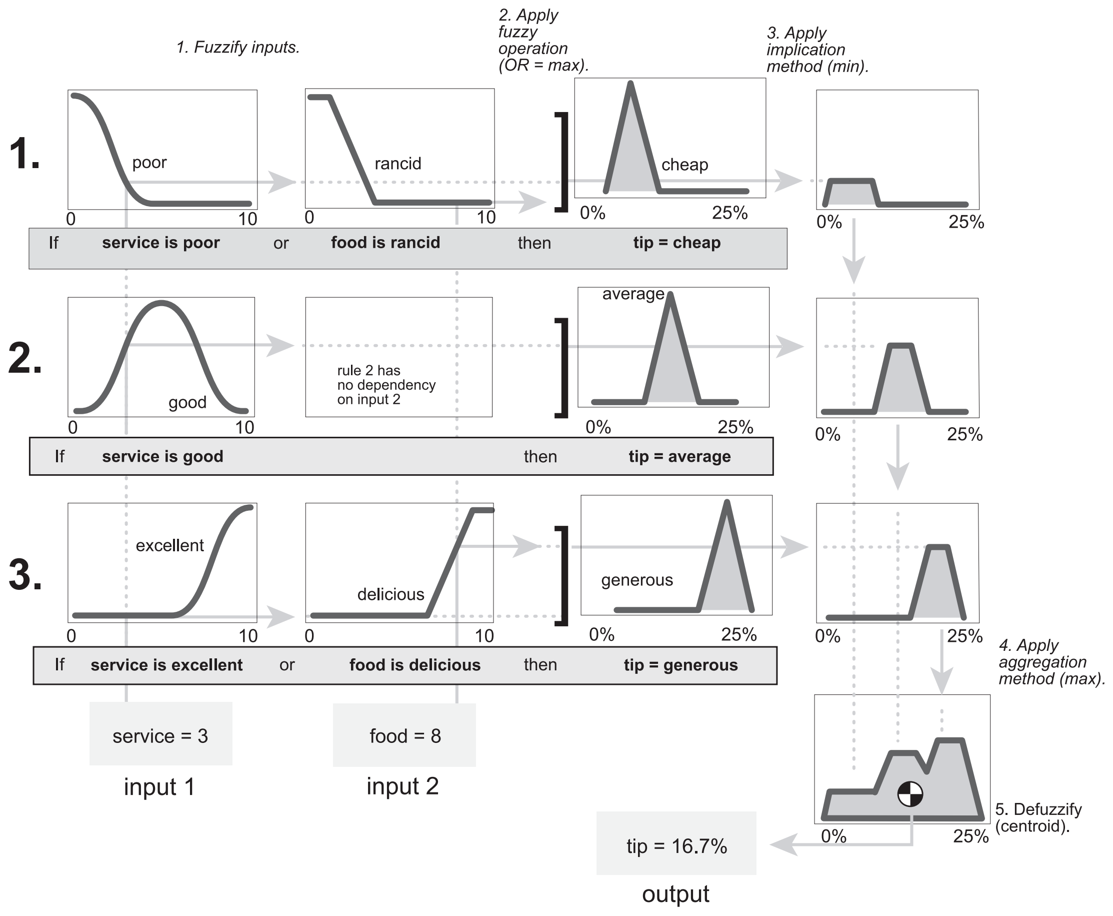

1. Fuzzify inputs.
2. Apply fuzzy operation (OR = max).
3. Apply implication method (min).
1.
poor
rancid
cheap
0
10
0
10
0%
25%
0%
25%
If service is poor or food is rancid then tip = cheap
2.
good
rule 2 has no dependency on input 2
average
If service is good then tip = average
3.
excellent
delicious
generous
If service is excellent or food is delicious then tip = generous
4. Apply aggregation method (max).
service = 3
input 1
food = 8
input 2
5. Defuzzify (centroid).
tip = 16.7%
output

Figure 4.8: The inference process of a fuzzy expert system [4].

Chapter 5

Static Policy Inference

In computer systems a security policy is essentially a set of rules specifying the way to secure a system for the present and the future. Forming a security policy is a challenging task: the system may be inherently complex with many potentially conflicting factors. Traditionally, security policies have had a strong tendency to encode a static view of risk and how it should be managed (most typically in a pessimistic or conservative way) [8]. Such an approach will not suffice for many dynamic systems which operate in highly uncertain, inherently risky environments. In many military operations, for example, we cannot expect to predict all possible situations.

Much security work is couched in terms of risk but in the real world there is benefit to be had from the use of such system. In military operations you may be prepared to risk a compromise of confidentiality if not doing so could cost lives. There is a need for operational flexibility in the decision-making processes, yet we cannot allow recklessness. Decisions need to be defensible and so must be made on some principled basis. It is very useful to be able to codify what a "principled basis" consists of since this serves to document "good practice" and facilitates its propagation.

The above discussion has been couched in terms of human decision-making processes. In some environments the required speed of system response may force an automated decision. Such automated decisions must also be made on a "principled basis", and some of these decisions may be very tricky. Automated support must be provided with decision strategies or rules to apply.

We investigate in this chapter how EAs can be used to extract security policy rules automatically from examples of decisions made in specified circumstances. This is an exercise in *policy inference.* The automation aspect of the inference is doubly useful: automated inference techniques can discover rules that humans would miss; and policies can be dynamically inferred as new examples of tricky decisions become available. Thus, the current policy can evolve to reflect the experience with the system. For example, if a human determines what the proper response should be based upon the information available, either in real-time or post facto, a conclusion is drawn that similar responses should be given under similar circumstances. Essentially, we attempt to partition the decision space such that each partition is associated with a response that is commensurate with the risk-benefit tradeoff for that partition.

In practice, different decision makers may come to different decisions in the same circumstances, particularly when the circumstances are tricky. They may use data that is not available to the inference engine to reach a decision, or else one may simply have a different appetite for risk. Therefore, the inference technique used must be able to handle sets of decision examples that do not seem entirely consistent.

This chapter documents some proof of concept experiments in using EAs to infer security policy models. All results suggest that the idea of inferring security policy using EAs is feasible. The rest of the chapter is organised as follows: Sections 5.1 and 5.2 present various experimentation details and results on using EAs to infer various security policy models from decision examples, namely the MLS Bell-LaPadula model, the budgetised MLS model and the Fuzzy MLS model. Section 5.4 summarises the main contributions and points out some potential avenues for future research. Finally, Section 5.5 concludes this chapter with a summary of results.

5.1 Experimentation on binary decision policies

Many security policies can be represented as a set of `IF <condition> THEN <decision>` rules. For example, the MLS Bell-LaPadula policy model presented in Section 3.1.1 can be represented as:

$$\begin{aligned}
&\texttt{IF } (access = read \wedge sl \geq ol \wedge sc \supseteq oc) \vee \\
&\quad (access = write \wedge sl \leq ol \wedge sc \subseteq oc) \\
&\texttt{THEN } decision = allow \\
&\texttt{IF } (access = read \wedge (sl < ol \vee sc \not\supseteq oc)) \vee \\
&\quad (access = write \wedge (sl > ol \vee sc \not\subseteq oc)) \\
&\texttt{THEN } decision = deny \qquad (5.1)
\end{aligned}$$

where sl and ol are subject and object sensitivity levels and sc and oc are subject and object category sets.

Here we choose to use Strongly Typed Genetic Programming (STGP) to discover an expression for the condition corresponding to each possible decision. Each tree based individual represents a candidate condition for an action. The leaf nodes of a tree are elements of the terminal set. These nodes can be either a variable that represents one of the decision-making factors or a constant. These nodes are joined together with operators which themselves are joined together with other operators recursively. The operators generally can be divided into three layers. The operators at the lowest layer evaluate the child nodes and return a value of the same type as them. Those of the next layer are logic relational operators such as $<$ or $\in$; these operators compare two typed values and return a Boolean value. These Boolean values are combined at the highest layer by means of logical operators such as $\wedge$ (AND), $\vee$ (OR) or $\neg$ (NOT). The root node in a tree must evaluate to a Boolean. Some well typed individuals that represent the conditions in the MLS Bell-LaPadula policy model are depicted in Figure 5.1. The leftmost individual resembles the condition of read access in the policy, i.e., $(sl > ol \vee sl = ol) \wedge (sc \supset oc \vee sc \equiv oc)$. The other two are logically equivalent individuals and deal only with the sensitivity aspect of the read access condition, i.e., $(sl > ol \vee sl = ol)$.

Using this representation, the security policy inference problem can be transformed into an n-class classification problem, in which n is the number of possible decisions in the policy. STGP is used to search for the <`condition`> part of each

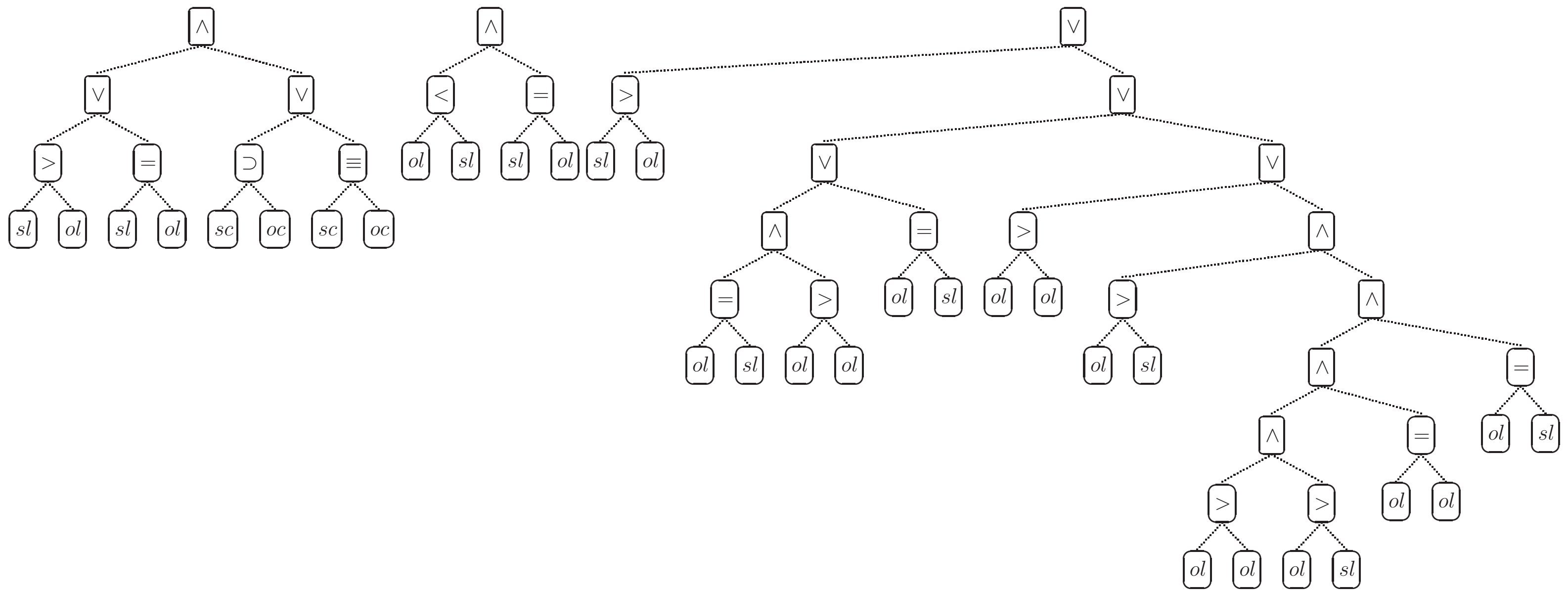


Figure 5.1: Some well typed individuals that represent the conditions in the MLS Bell-LaPadula policy model.

rule. Therefore, the number of STGP runs increases linearly with the policy size. Having said that, all runs can be carried out in parallel as they are independent from one another. In a single processor scenario, the binary decomposition method[1] can be employed to solve this problem in $n-1$ STGP runs.

The initial population is generated randomly using the ramp-half-and-half method popularised by Koza [88]. The individual fitness is computed using decision examples in a training set. Each example is represented by a vector of variables which corresponds to a set of decision-making factors and the decision itself. The fitness of the individual is based on the number of decision examples that it agrees with. In an ideal world, it might be desirable to match all examples. However, it is often the case in practice that there are a few poor decisions and many good decisions. The system might be expected to evolve a policy that agrees with the majority. 100% agreement is not essential. A lower degree of agreement may simply turn the spotlight on those specific individuals with decisions inconsistent with the inferred policy. In a sense, the fitness provides a measure of how well a candidate policy agrees with examined decisions, but also acts as anomaly detector. The accumulated total score after evaluating all examples becomes the fitness score of an individual.

After the fitness calculation stage, a new generation of individuals is produced with the use of evolutionary operators. Individuals with higher fitness scores have better chances to be selected to pass their "genes" (subtrees) to the next generation. Further crossover and mutation operators are applied probabilistically. The evolution process continues until an individual with a "high enough" fitness score is found or a preset number of generations have elapsed.

All GP based experiments in this chapter are carried out using ECJ[2] 18 [126]. The default parameter file in ECJ, i.e., `koza.params`, is used, unless otherwise specified. The GE based experiment presented in Section 4.3 is carried out using

[1]Binary decomposition method decomposes the n-class classification problem into $n-1$ binary classification problems. Let c_n represent class n. Then, the first classification problem is between c_1 and $c_1{}'$ ($c_1{}'$ is the complement class of c_1), the second one is between c_2 and $c_2{}'$ ($c_2{}' = c_1{}' - c_2$) and so on with the last one is between c_{n-1} and $c_{n-1}{}'$ ($c_{n-1}{}' = c_{n-2}{}' - c_{n-1} = c_n$). The algorithm is inherently sequential such that the n-th binary classification problem can only be solved after the $n-1$ previous problems are solved.

[2]ECJ is a popular evolutionary computation framework that includes the implementations of many popular EAs written in Java.

libGE[1] 0.26 [127] and GAlib[2] 2.4.7 [128] with all the parameter settings remaining as the default values unless specified otherwise.

5.1.1 Experiment 5.1: Partial MLS Bell-LaPadula policy

In the first experiment, we concentrate only on the "no read up" property (often known as the simple security property) of the MLS Bell-LaPadula model. This model is simple, unambiguous, and serves to demonstrate some interesting properties of our method of inference. For a read access, the model can be summarised as:

$$\begin{aligned}&\texttt{IF}\ \ sl \geq ol \wedge sc \supseteq oc\ \ \texttt{THEN}\ \ decision = allow\\&\texttt{IF}\ \ sl < ol \vee sc \not\supseteq oc\ \ \texttt{THEN}\ \ decision = deny\end{aligned} \tag{5.2}$$

where *sl* and *ol* are subject and object sensitivity levels and *sc* and *oc* are subject and object category sets. Since the *decision* is binary (either *allow* or *deny*), the GP algorithm only needs to be run once to search for the condition and the other condition can be simply obtained by logical negation. Here the condition for *allow* is chosen to be the learning target in this experiment.

The terminal set consists of four variables, namely *sl*, *ol*, *sc* and *oc* but no constant value. The *sl* and *ol* are positive integers and tagged with the type "sensitivity", for which three operators are defined: $=$, $<$ and $>$. The *sc* and *oc* are sets and given the type "category", for which three operators are defined: $\equiv$, $\subset$ and $\supset$. The $\neg$, $\leq$, $\geq$, $\subseteq$ and $\supseteq$ operators are intentionally omitted to make the search more difficult. Each category in *sc* or *oc* is represented by a positive integer. The target condition, $TC(sl, ol, sc, oc)$, to be learnt in this experiment is:

$$(sl > ol \vee sl = ol) \wedge (sc \supset oc \vee sc \equiv oc) \tag{5.3}$$

In each run of the experiment, the maximum value of sensitivity levels for *sl* and *ol*, SNS_{max}, and the total number of categories, CAT_{max}, are defined. Here

[1]libGE is a GE library written in C++. It was the only GE library available at the time the GE based experiment presented in Section 5.2.3 was carried out.

[2]GAlib is a popular GA library written in C++. It is used as the search algorithm in GE.

SNS_{max} and CAT_{max} are both set to be 5. We then randomly generate 100 examples to form the training set. Each example is a tuple consists of five attributes: *sl*, *ol*, *sc*, *oc* and *decision*. The values of *sl* and *ol* are randomly chosen from $\{1 \,..\, SNS_{max}\}$; the elements of *sc* and *oc* are randomly chosen from $\{1 \,..\, CAT_{max}\}$ such that the probability that any particular category is included in a specific set is 1/2; and the *decision* is set to be either 1 (*allow*) or 0 (*deny*) in accordance with (5.2). Thus, all decision examples here are correct as far as the MLS Bell-LaPadula policy model is concerned.

The fitness of an individual (candidate policy) is simply the sum of the matches between the decision made by the individual policy and the decision recorded in each example in the entire training set. Formally, let $d_{i,x}$ be the decision an individual i made for an example x with *True* as 1 and *False* as 0. The fitness of an individual i is then defined as follows:

$$f(i) = \sum_{\forall \ example \ x} (d_{i,x} \equiv decision_x) \tag{5.4}$$

Experimental results and evaluation

GP (and EAs in general) is stochastic in nature. The resulting policies depend on the random seed used. Thus, for each experimental setup here, 10 independent runs are carried out. Each run is initialised with a different random seed.

In all runs, logically equivalent conditions of TC in (5.3) can be learnt. Some examples of the evolved individuals are shown in Figure 5.2. The leftmost individual resembles TC with minimal possible tree size (number of nodes) whereas the other two individuals consist of some components that can be logically simplified, e.g., $ol > ol$ that appears in the rightmost individual is logically equivalent to *False*.

To investigate the robustness of this technique, the following changes are made to the experimental setup:

- Scaling up SNS_{max} and CAT_{max};
- Inclusion of "wrong" examples in the training set, i.e., inconsistent decision examples;

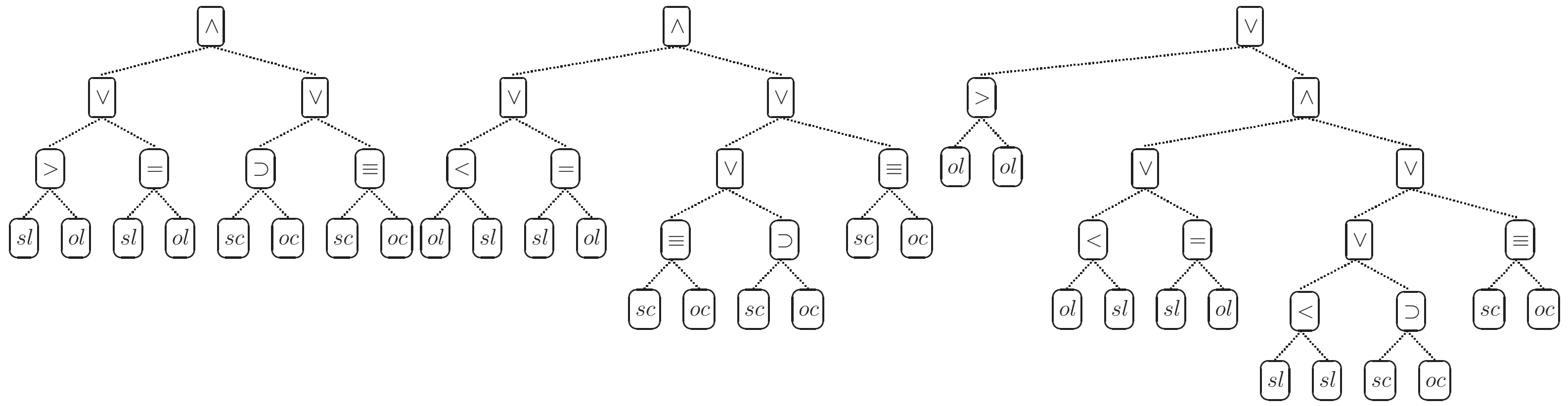


Figure 5.2: Some examples of inferred policies in Experiments 5.1.

- Changing other parameters including training set size, population size and tree height.

Scaling up SNS_{max} and CAT_{max} This experiment is repeated with six different settings of (SNS_{max}, CAT_{max}). These settings are $(10, 5)$, $(20, 5)$, $(30, 5)$, $(5, 10)$, $(5, 20)$ and $(5, 30)$. In the first three settings, in which SNS_{max} is scaled up to 30, logical equivalent conditions of TC can still be found. However, in the latter three settings, in which CAT_{max} is scaled to 30, only weaker conditions TC' are found. More precisely, the conditions learnt using the $(5, 10)$ setting are logically equivalent to $(sc \supset oc \vee sc \equiv oc)$; the conditions learnt using the $(5, 20)$ setting are either logically equivalent to $(sc \supset oc \vee sc \equiv oc)$ or simply $sc \supset oc$; and the conditions learnt using the $(5, 30)$ setting are logically equivalent to $sc \supset oc$.

Randomly generated category sets pose interesting problems from a training point of view. In our example generation process, the probability of randomly generating a pair (sc, oc) where $sc \equiv oc$ is small, $1/(2^{CAT_{max}})$ in fact. Thus, the expected number of category equality examples in a sample of size n is $n/(2^{CAT_{max}})$. Unless the system sees examples of how equality should be handled, it cannot be expected to infer how to handle it. Inference summarises rather than speculates. As usual, the training set characteristics are important. To validate this intuition, examples are manually created to cover the equality case and the experiment is repeated. The results agree with this intuition; logically equivalent conditions of TC are learnt for settings with CAT_{max} equal to 10, 20 and 30.

To further investigate the effect of training set coverage, Three runs with extreme settings are carried out. A training set with nine examples that cover all nine possible relationships between (sl, ol) and (sc, oc) in TC, namely $(>, =, <) \times (\supset, \equiv, \subset)$, is used. The learnt condition is logically equivalent to TC. At the other extreme, an experimental setup using all examples with *deny* decision ($decision = 0$) yields a logically equivalent condition of *False*. Conversely, if all examples in the training set are examples with *allow* decisions, then a logically equivalent condition of *True* is learnt. Thus, a mixture of correct *allow* and *deny* examples is required to evolve credible policies.

Inclusion of "wrong" examples This experiment is repeated with the introduction of wrong examples in the training set. We first started to introduce 1% "wrong" examples and then 10%, 20%, 25% and 30%. In all cases, the conditions learnt are similar to those learnt with all correct examples. This is because the search for the condition is guided by the fitness function which is defined as the number of matches between decisions made by an individual and the ones encoded in examples. In order to have maximum fitness, the search will tend to model the correct examples (which are in the majority) and choose to be inconsistent with the others. This is encouraging because 100% agreement is not the actual goal as mentioned earlier. Highlighting anomalous behaviours is also important[1].

Parameter changes This experiment is repeated with different parameter values. Firstly, the size of the training set is set to be 500 and 1000. In both cases, the conditions learnt are very similar to the ones learnt with only 100 examples.

Secondly, the experiment is repeated with various population sizes: 50, 100, 500 and 5000. When the population size is 500 or larger, logically equivalent conditions of TC are learnt, and there is no significant difference in terms of the number of generations required. However, the execution time per generation increases significantly because of the increase in the number of evolutionary operations and fitness evaluations performed. When the population size is set to be 50 or 100, the desired condition cannot be learnt in all cases. Manual investigation on the population reveals that the diversity in the population is lost in early generations, i.e., premature convergence in the population occurs. Many of these individuals get stuck at local optima such as $((sl > ol \vee sl = ol) \wedge (sc \supset oc))$ or simply $(sc \supset oc)$.

Lastly, we investigate the effect of tree size of each individual. The experiment is repeated with different maximum tree heights allowed. In each experiment, the maximum tree height is set to be one less than that of the previous experiment; the first experiment has maximum tree height of 17. We observe that the target

[1]The identification of anomalies is not the focus of the research, but the ability to identify such cases as a consequence of attempting inference is clearly of some potential use. They either point to errant behaviour, or else identify difficult situations where information outside the model might have been useful in the decision-making process.

condition cannot be learnt if the tree height is less than 4, which is the minimum height to represent TC.

5.1.2 Experiment 5.2: Full MLS Bell-LaPadula policy

We extend Experiment 5.1 to evolve the MLS Bell-LaPadula policy model as a whole for both read access and write access. The rules that represent this policy model is shown in (5.1) and are restated as follows for ease of reference:

$$\begin{aligned}
&\texttt{IF } (access = read \wedge sl \geq ol \wedge sc \supseteq oc) \vee \\
&\quad (access = write \wedge sl \leq ol \wedge sc \subseteq oc) \\
&\texttt{THEN } decision = allow \\
&\texttt{IF } (access = read \wedge (sl < ol \vee sc \not\supseteq oc)) \vee \\
&\quad (access = write \wedge (sl > ol \vee sc \not\subseteq oc)) \\
&\texttt{THEN } decision = deny
\end{aligned}$$

where *sl* and *ol* are subject and object sensitivity levels and *sc* and *oc* are subject and object category sets.

The experimental setup remains to be the same except for the following. Firstly, a new type "access" is introduced to the type set and the terminal set is expanded to include three new terminals of this type: *access*, *read* and *write*. Secondly, the function set is extended to include $\leq$, $\geq$, $\subseteq$ and $\supseteq$ operators. Thirdly, the training set size is increased to 500 randomly generated examples with the equality cases guaranteed as described in Section 5.1.1. All other parameter settings including the fitness function remain the same as before.

Here the target condition, $TC(access, sl, ol, sc, oc)$, to be learnt in this experiment is:

$$\begin{aligned}
&(access = read \wedge sl \geq ol \wedge sc \supseteq oc) \vee \\
&(access = write \wedge sl \leq ol \wedge sc \subseteq oc)
\end{aligned} \tag{5.5}$$

Experimental results and evaluation

As in Experiment 5.1, 10 independent runs of this experiment are carried out. Each is initialised with a different random seed. In all runs, logically equivalent conditions of TC in (5.5) can be learnt. The evolved individuals do not always resemble TC with minimal tree size. Some of the subtrees in the evolved individuals are redundant and may be simplified to simpler logical expressions. We then investigate the robustness of this inference technique as before by scaling up SNS_{max} and CAT_{max}, inclusion of "wrong" examples in the training set and changing of parameter values.

Scaling up SNS_{max} and CAT_{max} The SNS_{max} and CAT_{max} are scaled up as in Experiment 5.1 using six different settings of (SNS_{max}, CAT_{max}): $(10, 5)$, $(20, 5)$, $(30, 5)$, $(5, 10)$, $(5, 20)$ and $(5, 30)$. As the training set used covers the category equality cases, logically equivalent conditions of TC in (5.5) are learnt in all cases.

Inclusion of "wrong" examples Here we introduced 1%, 10%, 20%, 25% and 30% "wrong" examples. In all but the 30% cases, the conditions learnt are similar to those learnt with all correct examples. In the 30% case, the number of successful runs decreases to 7 out of 10 runs. Investigation of the best individuals in unsuccessful runs shows that these individuals generally represent conditions that are slightly weaker or stronger than TC. Some of these individuals after manual logical simplification are shown as follows:

$$\begin{aligned}&(access = read \wedge sl \geq ol \wedge sc \supseteq oc) \vee \\ &(access = write \wedge sl \leq ol \wedge sc \subseteq oc) \vee \\ &(sc = oc)\end{aligned} \tag{5.6}$$

$$\begin{aligned}&(access = read \wedge sl > ol \wedge sc \supset oc) \vee \\ &(access = write \wedge sl < ol \wedge sc \subseteq oc)\end{aligned} \tag{5.7}$$

Although these conditions are not logically equivalent to TC, these conditions agree with more than 90% of the examples in the training set. In other words, the models that these individuals represent are very good approximations to the MLS Bell-LaPadula policy model.

Parameter changes This experiment is repeated with different parameter values as described in Section 5.1.1. In all cases, similar results are found. The change of training set size does not affect the results. If the population size is set to be too small, i.e., 50 or 100, the population may prematurely converge and get stuck at local optima. The only difference here is that the minimum tree heights required is increased to 5, which is the minimum tree height required to represent the more complicated TC defined in this experiment.

5.1.3 Experiment 5.3: Budgetised MLS policy

Overall, it would seem that the approach taken is easily capable of summarising or distilling a policy from a given set of examples, even in the presence of "noise". However, the fitness function used (number of agreements) is very blunt. All decisions are considered equal. This is not the case in practice. Granting an uncleared user access to a top secret document is much riskier than granting him access to a secret document. We need to investigate the approach from a more genuinely risk based perspective. In this experiment, we designed a simple risk based policy with intuition drawn from [2, 8]. For a read access, the budgetised MLS policy is as follows:

$$\begin{aligned}
&\texttt{IF}\ \ (\mathrm{pos}(ol - sl) + \#(oc \setminus sc)) \leq \mathit{offer}\ \ \texttt{THEN}\ \ \mathit{decision} = \mathit{allow} \\
&\texttt{IF}\ \ (\mathrm{pos}(ol - sl) + \#(oc \setminus sc)) > \mathit{offer}\ \ \texttt{THEN}\ \ \mathit{decision} = \mathit{deny}
\end{aligned} \tag{5.8}$$

where $\mathrm{pos}(x)$ returns $\max(x, 0)$; $x \setminus y$ is the set difference between set x and set y; $\#(x)$ is the cardinality of the set x and *offer* is the amount the requester is willing to pay for the requested access. For this experiment, the target condition, $TC(sl, ol, sc, oc, \mathit{offer})$, becomes:

$$\mathrm{pos}(ol - sl) + \#(oc \setminus sc) \leq \mathit{offer} \tag{5.9}$$

Note that there are two cost components here. We have decided, for the sake of simplicity, to equate the cost of a single band difference in sensitivity level with the cost of a single set difference in category. (Other weightings are clearly possible.)

The experimental setup is similar to Experiment 5.2 except for the following. All the terminals of the "access" type are removed from terminal set. A new type, "numeric", is added to the type set. A new terminal *offer* of this type is added to terminal set and eight new operators that return values of this type are added to function set. These operators are pos(x), #(x), $\backslash(x, y)$, $+$, $-$, $\times$, $\div$[1], exp(x).

Experimental results and evaluation

As in previous experiments, 10 independent runs of the experiment are carried out. Each run is initialised with a different random seed. In all cases, the results show that the logically equivalent conditions of TC in (5.9) can be learnt in all cases. The investigation on the robustness of this inference technique by scaling up SNS_{max} and CAT_{max}, inclusion of "wrong" examples in the training set and changing of parameter values produce similar results. The only difference is that the minimum tree size required now is increased to 8 because of the higher complexity of the TC.

5.2 Experimentation on multi-decision policies

Up to this point, we have only considered binary decision policies. In this experiment, we attempt to infer policies that have more than two decisions. Here the Fuzzy MLS security policy model presented in Section 3.4.1 is used as the reference model. This policy uses the risk based rationale of the MLS Bell-LaPadula policy to compute a quantified risk estimate by quantifying the "gap" between a subject's label and an object's label in an MLS system. Quantified risk estimates are numbers and therefore can be used to build a risk scale (refer to Figure 3.2).

[1] $x \div y \;=\; \begin{cases} x/y & \text{if } y \neq 0 \\ 1 & \text{otherwise} \end{cases}$

This risk scale is divided into multiple bands. Each band is associated with a decision. The risk in the bottom band is considered low enough so the access decision is simply *allow* whereas the risk in the top band is considered too high so the decision is *deny*. Each band between the top and bottom is associated with a *allow decision with different risk mitigation measures*.

The Fuzzy MLS model defines risk as the expected value of damage caused by unauthorised disclosure of information in (3.1). This definition is restated as follows for ease of reference:

$$\text{risk} = (\text{value of damage, } V) \times (\text{probability of incurring the damage, } P)$$

The value of damage is estimated from the sensitivity level of the object and is defined to be a^{ol}. The probability of incurring the damage is estimated by quantifying two "gaps": one between the sensitivity levels of the subject and the object, and the other between the category sets of the subject and the object. For simplicity, this experiment only looks at the sensitivity levels and assumes the category sets are the same[1]. The probability of incurring the damage is thus defined as a sigmoid function in (3.5). This is restated as follows:

$$P(sl, ol) = \frac{1}{1 + \exp((-k) \times (TI(sl, ol) - n))}$$

where $TI(sl, ol)$ is called the *temptation index* which indicates how much the subject with sensitivity of *sl* is tempted to leak information with sensitivity level of *ol*. It is defined in (3.4) and is restated as follows:

$$TI(sl, ol) = \frac{a^{-(sl-ol)}}{m - ol}$$

The intuition for $P(sl, ol)$ and $TI(sl, ol)$ can be found in Section 3.4.1. In our experiments, the settings of $a = 10$, $k = 3$, $n = 4$ and $m = 11$ are used here. These are the values used in [2].

Long standing convention groups risk by "order of magnitude". Thus, if we want to avail ourselves of a linear banding scale, taking the logarithm of the risk

[1]Therefore the gap between category sets is 0.

values seems in order. The following formula is defined to map a risk number to a risk band:

$$band(risk(sl, ol)) = \max(\min(\lfloor \log_a(risk(sl, ol)) \rfloor,\ a-1),\ 0) \qquad (5.10)$$

where the function $risk(sl, ol)$ is defined in (3.1). Base-a logarithm is used to compute the order of magnitude of risk as the band number. Since each band is associated with a decision, a risk band number computed using (5.10) represents a possible decision in the policy.

To generate the data required for training and testing purposes, SNS_{max} is set to be 10. A data set is generated, consisting of the 100 possible $(sl, ol, band)$ examples, where sl and ol are integers in $[0, 9]$ and $band$ is calculated using (5.10). In other words, all the examples used are assumed to be correct. As the data are limited, the leave-one-out cross validation evaluation method (LOOCV) is used to evaluate the performance.

In the traditional hold-out evaluation method, the data set is separated into two subsets: the training set and the testing set. The model is learnt using the data in the training set and then is evaluated using the data in the testing set. This method has the merit of being computationally cheap. However, it wastes a lot of data and its evaluation can have a high variance depending on how the data are split.

k-fold cross validation can be used to improve the hold-out method. Instead of two subsets, the data set is split into k subsets. The hold-out method is then repeated k times, each using data in $k-1$ subsets as the training set and the data in the remaining one as the testing set. The evaluation is done using the average performance across all k trials. The method is less sensitive to the way the data are split and therefore results in a smaller variance in the evaluation. However, the training process has to be repeated k times, which can be costly. LOOCV takes k-fold cross validation to its extreme, with k equal to the number of examples in the data set. The evaluation obtained using LOOCV is very good but extremely expensive to compute, especially in EAs.

To evaluate the performance of the policies inferred, each experiment is repeated 100 times with a different random seed. The performance is evaluated by two criteria:

1. The median LOOCV error rate of the best individuals in the 100 runs. The best individual in a run is the one with the lowest error rate in the last generation of the run. The median is used instead of the mean as it does not assume that the error rate distribution will have a suitably normal distribution. Confidence intervals based on the standard deviation of the mean are no longer valid. The 95% confidence interval of the median is calculated using the Thompson-Savur formula presented in [129] instead.

2. The median of the average distances between all the predicted bands of the best individuals and the target bands encoded in all the examples in the data set in the 100 runs.

These measurements indicate the quality of the security policy that can be learnt in each experiment.

5.2.1 Experiment 5.4: Rule based approach

In this experiment, we continue to employ the same view as in previous experiments: a security policy is a set of `IF-THEN` rules. As the risk scale is divided into 10 bands, each band is associated with a decision action, 10 STGP runs are required to search for conditions for all the bands. We will seek for each risk band j, a target condition $TC_j(sl, ol)$ that returns *allow* for examples in risk band j and *deny* for those not in risk band j. Thus, $TC_j(sl, ol)$ is a membership classifier for band j.

We observe that the reference model for calculating the risk is actually rather complicated. We might reasonably seek to evolve sets of membership classifiers that are nearly correct, e.g., very good at classifying examples in the appropriate band, but which occasionally consider examples from near-by bands as being within the band. Examples from bands significantly different to the one at hand should be rejected from membership by the membership classifier.

To give a good chance of classifying appropriately, we adopt a "high watermark" approach. The final band classification is the highest band j for which the corresponding membership classifier TC_j returns *allow*. Formally, if $TC_j \equiv allow$

and $TC_k \equiv$ *allow* and $j > k$, then band j instead of band k is used. Additionally, if there is no TC evaluated to *allow*, the highest band is used again for security concerns. In a future run-time deployment of our inference approach, it would be possible to involve human interaction. An alert can be given to the security administrator and the administrator can decide which band an input should be mapped to. This decision can then be used as a new training example.

Here we use only float typed nodes to avoid the overhead in checking the type conformance of the nodes in STGP. We assume that an individual represents *allow* if its evaluated value is 1 and *deny* if its evaluated value is 0. Individuals with any other evaluated value are assumed to be invalid and therefore are assigned with the lowest possible fitness to increase their chances of being eliminated in the evolution process. The terminal set consists of *sl*, *ol* and Ephemeral Random Constants (ERCs)[1], which take real values in $[-10, 10]$. The function set consists of $+$, $-$, $\times$, $\div$, protectedlog$_{10}(x)$[2], $\exp(x)$, $\mathrm{pow}(x, y)$, $\max(x, y)$, $\min(x, y)$, $\mathrm{ceil}(x)$, $\mathrm{floor}(x)$, $\sin(x)$ and $\cos(x)$.

Two principles are used to determine the fitness score for a decision made by an individual. For an example x and an individual i, if i evaluates x to be in band j, then:

- For a correct decision, *reward more* the higher the risk band, i.e., reward higher j more than lower j. (We care more about higher risk bands.)
- For an incorrect decision, *punish more* the more the decision deviates from the target; i.e., punish more as $|band_x - j|$ becomes larger, where $band_x$ is the decision encoded in the example x. Additionally, punish *under-estimation* of a risk band more than over-estimation of it; i.e., punish more if $band_x > j$.

Based on these principles, let $d_{i,x}$ be the decision made by an individual i for example x. Then, the fitness function of an individual in the run that search for

[1]An ERC is a constant that its value is randomly generated during its creation.

[2]$\mathrm{protectedlog}_{10}(x) = \begin{cases} \log_{10}(|x|) & \text{if } x \neq 0 \\ 0 & \text{otherwise} \end{cases}$

the condition of band j, TC_j, is defined as follows:

$$\begin{aligned} f_j(i) = & \sum_{\{x|band_x \equiv j\}} w_{tp}\{d_{i,x} \equiv allow\} + \sum_{\{x|band_x \not\equiv j\}} w_{tn}\{d_{i,x} \equiv deny\} \\ & - \sum_{\{x|band_x \not\equiv j\}} w_{fp}\{d_{i,x} \equiv allow\} - \sum_{\{x|band_x \equiv j\}} w_{fn}\{d_{i,x} \equiv deny\} \end{aligned} \tag{5.11}$$

where

$$\begin{aligned} w_{tp} &= j+1, \\ w_{tn} &= (j+1)/10, \\ w_{fp} &= \begin{cases} band_x - j & \text{if } band_x > j, \\ (j - band_x)/2 & \text{if } band_x < j, \end{cases} \\ w_{fn} &= j+1 \end{aligned}$$

5.2.2 Experiment 5.5: Regression based approach

In this experiment, we take an alternative view on a policy. We view a policy as a function that maps a set of decision-making factors to a decision. In the Fuzzy MLS model, this mapping function is the composition band and risk function in (5.10). GP is used to search for an equivalent function of this composition function. This is essentially an exercise of symbolic regression based upon decision examples.

The experimental setup is very similar to Experiment 5.4 except for the fitness function used. Here the fitness function is defined to be the reciprocal of one plus the sum of squared errors between the value an individual is evaluated to and each of the correct band encoded in the example set[1]. Formally, let j_x be the value an individual i evaluates an example x to, the fitness of an individual i, is defined as follows:

$$f(i) = \frac{1}{1 + \sum_{\forall\ example\ x} (band_x - j_x)^2} \tag{5.12}$$

[1]One is added to the sum of squared errors before inversion to avoid division by zero.

To determine the band of a particular (sl, ol) pair, the output value of the learnt policy is rounded to the nearest integer value. The learnt policy might not be perfect; sometimes the policy may map a particular (sl, ol) pair to a value that is out of band range. To overcome this, we assume that all out-of-range values represent the highest band, i.e., any output value that is not in the range $[0, 9]$ is assumed to be 9. This is consistent with the usual attitude to security which favours over-estimation of risk rather than under-estimation.

5.2.3 Experiment 5.6: Grammatical evolution

In this experiment, Grammatical Evolution (GE) is used as an example to show that other EAs can also be used to search for security policies. Experiment 5.5 presented in Section 5.2.2 is repeated here with minimal changes. A policy is viewed as a function that maps a set of decision-making factors to a decision and GE is used to search for this function. The BNF grammar that describes the search space is defined as follows:

$$\begin{aligned}
N &= \{\texttt{<expr>}, \texttt{<sub\_expr>}, \texttt{<unary\_op>}, \texttt{<binary\_op>}, \texttt{<var>}, \texttt{<const>}, \texttt{<digit>}\} \\
T &= \{\sin, \cos, \exp, \text{protectedlog}_{10}, \text{ceil}, \text{floor}, \min, \max, -, +, -, \times, \div, -, sl, ol, \\
&\qquad ., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \\
S &= \texttt{<expr>}
\end{aligned}$$

and P consists of a set of production rules as follows:

```
<expr>       ::= <unary_op>(<expr>) | <binary_op>(<expr>, <expr>)
              |  <sub_expr>
<sub_expr>   ::= <var> | <const>
<unary_op>   ::= sin | cos | exp | protectedlog | ceil | floor | −
<binary_op>  ::= + | − | × | ÷ | min | max
<var>        ::= sl | ol
<const>      ::= <digit>.<digit><digit>
<digit>      ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

The primitive operators are wrapped as a function call to prevent any bias from being introduced among the operators. Instead of using ERCs, the `<const>`

and <digit> rules are used to generate random constants in the range $(-10, 10)$. Generating random constants in such a fashion enables random numbers to partake in the evolutionary process.

The evolutionary operators used are crossover and mutation with probabilities of 0.9 and 0.01 respectively. Two different implementations of crossover, namely one-point crossover and effective crossover, are investigated. One-point crossover randomly chooses a point on each of the two selected individuals and swaps all data beyond that point, whereas effective crossover restricts the chosen crossover point in the range of effective length of each individual (the portion of an individual that is actually used to select the rules).

Two different ways of initialising the population of individuals are investigated: random initialisation and sensible initialisation [92]. The former takes two parameters, $length_{min}$ and $length_{max}$, and produces a population of individuals with lengths evenly distributed over a range $[length_{min}, length_{max}]$. The settings of $length_{min} = 15$ and $length_{max} = 25$ are used here. The latter method takes two parameters, $height_{min}$ and $height_{max}$, and produces a population of individuals that correspond to programs with derivative trees of the size between a range $[height_{min}, height_{max}]$. The settings of $height_{min} = 1$ and $height_{max} = 10$ are used here.

The steady state genetic algorithm is used as the search algorithm with 25% replacement rate (the percentage of the population that is replaced at each iteration). The roulette wheel selection scheme is used as the selection scheme.

As in Experiment 5.5, to determine the band of a particular (sl, ol) pair, the output value of the learnt policy is rounded to the nearest integer value and all out-of-range output values of the learnt policy are assumed to be 9.

5.2.4 Experiment 5.7: Fuzzy set ensemble

In this experiment, we aim to provide some degree of smoothing to our search space by adopting a fuzzy inspired approach. As opposed to traditional approaches whereby EAs are often used in searching for the optimal weighting among some predefined fuzzy rules, we view each possible risk band as a fuzzy set and use EAs to learn the fuzzy membership function for each of the bands.

Once these functions have been learnt, the most appropriate risk band for a given input (sl, ol) can be determined by feeding the input to all learnt functions and aggregating all the outputs using a predefined voting based defuzzification mechanism.

This approach has several advantages. The estimation of a risk band is likely to be more accurate with the use of multiple functions. For example, if each membership value indicates that an input should be mapped to band 5, then with very high confidence we can say the input belongs to band 5. This is analogous to drawing the final conclusion by examining the input from various different perspectives, which is more likely to be accurate than examining the input from one perspective. Additionally, as will become clear in Section 5.2.4, the learning process of each fuzzy membership function is essentially a *curve fitting* exercise, which is naturally more tolerant to incomplete coverage in the training set because it uses *interpolation* and *extrapolation* to compensate for the "missing points". It is also more resilient to the outliers in the training set.

Fuzzification

To learn the fuzzy membership function for band j, $M_j(sl, ol)$, we first define several values, each represents the value M_j should map to for an input with a particular band. In our case, 10 values are defined (because $a = 10$). Essentially, these values define the shape and location of the M_j function. We then use EAs to search for a curve that best fits the 10 values, using all the examples in the training set.

To define the 10 values for each M_j, we choose to use values between $[-1, 1]$ with 0 representing the value for band j. For example, the values for $M_5(sl, ol)$ that correspond to each band, starting from band 0, can be defined as:

$$[-0.5, -0.4, -0.3, -0.2, -0.1,\ 0.0,\ 0.1,\ 0.2,\ 0.3,\ 0.4]$$

The sign of the value allows information about the direction of the band it represents to be encoded (positive implies it is greater than band j while negative implies it is less than band j) and the magnitude of the value increases with the distance between the band it represents and band j. Note that risk band j always has the value of 0. This information is useful in designing the defuzzification mechanism. The 10 values for other $M_{j\neq 5}$ can be defined similarly.

Two different predefined sets of values are setup to validate this concept. In the first setup, the 10 values are defined using the following function:

$$M_j[k] \equiv (k - j)/10 \tag{5.13}$$

where k is the index of each element, having an integer value in the range $[0, 9]$. This is like mapping a traditional triangular fuzzy membership function, which has the range $[0, 1]$ and $M_j[j] \equiv 1$ as the tip of the triangle, to a straight-line membership function with the range $[-1, 1]$ and $M_j[j] \equiv 0$. Figure 5.3 shows the values for all 10 bands using (5.13). In the second setup, the bell-shaped Gaussian distribution curve is used in a similar fashion and the values for all 10 bands are shown in Figure 5.4.

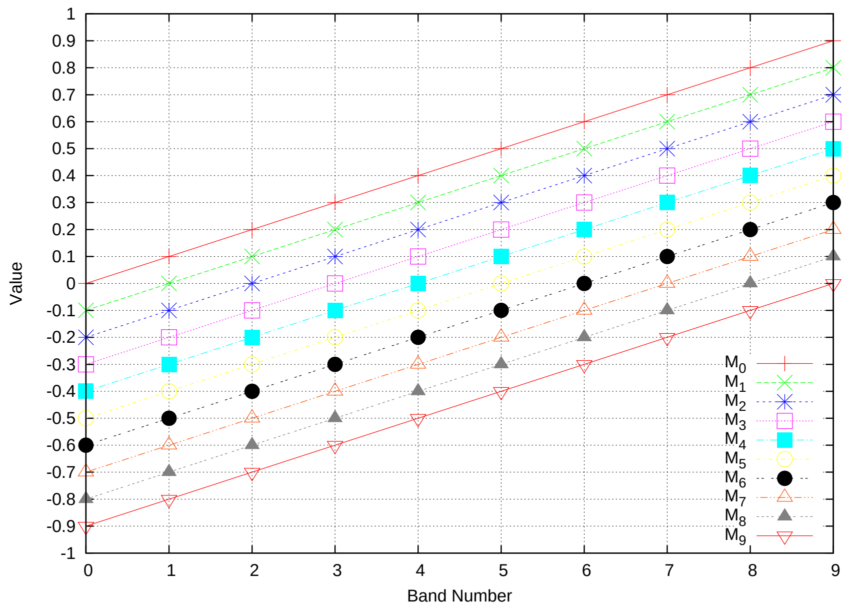


Figure 5.3: The predefined values for each M_j based on the triangular fuzzy membership function.

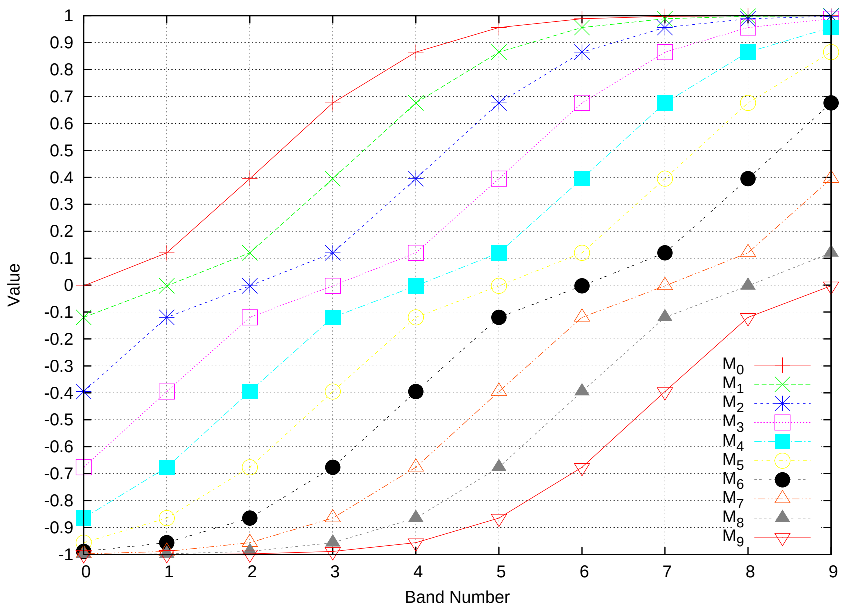


Figure 5.4: The predefined values for each M_j based on the Gaussian distribution curve.

Individual representation and fitness evaluation

The same terminal and function sets in Experiment 5.4 are used here. It follows that the structure of the individuals remains similar. However, what each individual represents varies from one experiment to another. Here each individual resembles the membership function of a particular band whereas each individual in Experiment 5.4 resembles the Boolean condition of a particular band and each individual in Experiments 5.5 and 5.6 resembles a function corresponding to the policy as a whole.

The fitness function is the sum of squared differences between the predicted membership and the predefined value for each band. More formally, let $M'_{j,i}$ represent an individual i in the search for the membership function for band j, the individual fitness is defined as follows:

$$f_j(i) = \frac{1}{1 + \sum_{\forall\ example\ x} \left(M'_{j,i}(sl_x, ol_x) - M_j[band_x]\right)^2} \tag{5.14}$$

Defuzzification

After all 10 membership functions are learnt and feeding an input $x \equiv (sl_x, ol_x)$ to these functions, we need a *defuzzification* mechanism to map all 10 values returned by these functions to a risk band number. Three voting based defuzzification algorithms which use the information about the direction and distance to determine the target band have been designed. The target band is predicted by adding the estimated distance (calculated from the output value and rounded to an integer) to the band of the membership function. Votes are then given to a small range of bands around the predicted target band. The difference between these algorithms lies in the weighting of the votes.

Algorithm 5.1 forces the predicted target band to be in the range $[0, 9]$. Votes are then given in the following ways: 3 votes to the predicted target band, 2 votes to the nearest neighbour band and 1 vote to the second nearest neighbour band, i.e., the nearest neighbour band on the other side of the predicted target band.

Algorithm 5.2 is very similar to Algorithm 5.1, except for the following. Firstly, the predicted target band is no longer forced to be in the range $[0, 9]$. Secondly, votes with equal weight are given to the nearest neighbour bands on both sides of the predicted target band. Thirdly, the weighting of the votes is inspired from the standard Gaussian Distribution, in which 0.3829250 vote is given to the predicted target band, 0.2417300 vote is given to the nearest neighbour band on each side and 0.0605975 vote is given to the second nearest neighbour band on each side.

Algorithm 5.3 attempts to simplify these algorithms. The estimated distance is first calculated as before but it is not rounded to an integer. This algorithm then gives votes to the upper bound and lower bound of the predicted target band, with the weighting of these votes inversely proportional to the distance.

5.2.5 Experimental results and evaluation

The results of Experiments 5.4– 5.7 are summarised in Table 5.1. In all GP based experiments (except for the rule based approach in Experiment 5.4), the results suggest that the target policy can be learnt reasonably well. (In a multi-class classification problem, assuming the examples used are evenly distributed across

```
initialise an array $v[10]$ with all elements set to $0$;
forall example $x$ do
    forall band $j$ do
        if $M_j(sl_x, ol_x) > 0.05$ then
            $p \leftarrow j + M_j(x) * 10$;
            $k \leftarrow min(\lfloor p + 0.5 \rfloor, 9)$;
            $v[k] \leftarrow v[k] + 3$;
            if $k > p$ then
                $v[k-1] \leftarrow v[k-1] + 2$;
                $v[k+1] \leftarrow v[k+1] + 1$;
            else
                $v[k-1] \leftarrow v[k-1] + 1$;
                $v[k+1] \leftarrow v[k+1] + 2$;
        else if $M_j(sl_x, ol_x) < -0.05$ then
            $p \leftarrow j + M_j(x) * 10$;
            $k \leftarrow max(\lceil p - 0.5 \rceil, 0)$;
            $v[k] \leftarrow v[k] + 3$;
            if $k < p$ then
                $v[k+1] \leftarrow v[k+1] + 2$;
                $v[k-1] \leftarrow v[k-1] + 1$;
            else
                $v[k+1] \leftarrow v[k+1] + 1$;
                $v[k-1] \leftarrow v[k-1] + 2$;
        else
            $v[j] \leftarrow v[j] + 3$;
            $v[j+1] \leftarrow v[j+1] + 1$;
            $v[j-1] \leftarrow v[j-1] + 1$;
choose $v[i]$ with the maximum value;
output $i$ as the risk band number;
```

Algorithm 5.1: A direction and distance based defuzzification.

```
*initialise an array* $v[10]$ *with all elements set to* $0$;
$GaussianConst = \{0.382925,\ 0.241730,\ 0.0605975\}$;
**forall** *example* $x$ **do**
    **forall** *band* $j$ **do**
        $p \leftarrow j + M_j(sl_x, ol_x) * 10$;
        $k \leftarrow \lfloor p + 0.5 \rfloor$;
        $v[k] \leftarrow v[k] + 3$;
        **if** $0 \leq k \leq 9$ **then**
            $v[k] \leftarrow v[k] + GaussianConst[0]$;
        **if** $0 \leq k + 1 \leq 9$ **then**
            $v[k+1] \leftarrow v[k+1] + GaussianConst[1]$;
        **else if** $0 \leq k \leq 9$ **then**
            $v[k] \leftarrow v[k] + GaussianConst[1]$;
        **if** $0 \leq k - 1 \leq 9$ **then**
            $v[k-1] \leftarrow v[k-1] + GaussianConst[1]$;
        **else if** $0 \leq k \leq 9$ **then**
            $v[k] \leftarrow v[k] + GaussianConst[1]$;
        **if** $0 \leq k + 2 \leq 9$ **then**
            $v[k+2] \leftarrow v[k+2] + GaussianConst[2]$;
        **else if** $0 \leq k + 1 \leq 9$ **then**
            $v[k+1] \leftarrow v[k+1] + GaussianConst[2]$;
        **else if** $0 \leq k \leq 9$ **then**
            $v[k] \leftarrow v[k] + GaussianConst[2]$;
        **if** $0 \leq k - 2 \leq 9$ **then**
            $v[k-2] \leftarrow v[k-2] + GaussianConst[2]$;
        **else if** $0 \leq k - 1 \leq 9$ **then**
            $v[k-1] \leftarrow v[k-1] + GaussianConst[2]$;
        **else if** $0 \leq k \leq 9$ **then**
            $v[k] \leftarrow v[k] + GaussianConst[2]$;
*choose* $v[i]$ *with the maximum value*;
*output* $i$ *as the risk band number*;
```

Algorithm 5.2: A direction and distance based defuzzification.

```
initialise an array $v[10]$ with all elements set to $0$;
forall example $x$ do
    forall band $j$ do
        $p \leftarrow j + M_j(sl_x, ol_x) * 10$;
        $j \leftarrow \lfloor p \rfloor$;
        $k \leftarrow \lceil p \rceil$;
        if $0 \leq j \leq 9$ and $0 \leq k \leq 9$ then
            $v[j] \leftarrow v[j] + k - p$;
            $v[k] \leftarrow v[k] + p - j$;
        else if $0 \leq k \leq 9$ then
            $v[k] \leftarrow v[k] + 1$;
        else if $0 \leq j \leq 9$ then
            $v[j] \leftarrow v[j] + 1$;
choose $v[i]$ with the maximum value;
output $i$ as the risk band number;
```

Algorithm 5.3: A direction and distance based defuzzification.

all classes, the error rate of a random classifier is $\frac{n-1}{n}$, where n is the number of classes. In this case, $n = 10$ and therefore the error rate of a random classifier is 0.9.) Moreover, the medians of the average distances between all the predicted bands and the target bands encoded in all the examples in these experiments are kept to be around 0.2 band.

To investigate why the policies inferred using the rule based approach perform poorly, we manually investigate the outputs of these policies. We observe that there are many unusual cases such that some (sl, ol) pairs with $(high, low)$ values are mapped to band 9. This is found to be caused by the use of "high watermark" approach in resolving the policy outputs. This pessimistic policy resolution mechanism degrades the performance significantly.

The experiments using the fuzzy set ensemble based approaches consistently perform very well in all six settings. The triangular fuzzification algorithm performs slightly better than the Gaussian fuzzification algorithm in all cases. This could be due to the fact that the defuzzification algorithm makes the assumption that the distance from the target increases linearly with respect to the membership value. This is not the case in Gaussian fuzzification algorithm. Possible further work here would be to design a compatible defuzzification algorithm using

EA	Experiment	Median Error Rate with 95% Confidence Interval / %	Median Distance with 95% Confidence Interval / band
GP	`IF-THEN` Rules	0.320 (0.310, 0.330)	1.420 (1.350, 1.470)
	Regression with mean square error	0.160 (0.150, 0.170)	0.210 (0.199, 0.240)
	Triangular fuzzification with defuzzifier 1	0.160 (0.149, 0.160)	0.180 (0.170, 0.190)
	Gaussian fuzzification with defuzzifier 1	0.160 (0.160, 0.170)	0.260 (0.249, 0.280)
	Triangular fuzzification with defuzzifier 2	0.140 (0.130, 0.150)	0.170 (0.160, 0.180)
	Gaussian fuzzification with defuzzifier 2	0.150 (0.140, 0.151)	0.230 (0.210, 0.240)
	Triangular fuzzification with defuzzifier 3	0.150 (0.140, 0.160)	0.190 (0.180, 0.190)
	Gaussian fuzzification with defuzzifier 3	0.150 (0.139, 0.160)	0.240 (0.230, 0.260)
GE	Random initialisation and one-point crossover	0.440 (0.430, 0.451)	0.850 (0.819, 0.881)
	Random initialisation and effective crossover	0.440 (0.430, 0.450)	0.775 (0.740, 0.820)
	Sensible initialisation and one-point crossover	0.425 (0.410, 0.440)	0.830 (0.787, 0.860)
	Sensible initialisation and effective crossover	0.420 (0.410, 0.440)	0.780 (0.760, 0.800)

Table 5.1: The experimental result summary of the inferred policies using GP and GE.

Approach	Example of the optimal policy inferred
GP	max(sin(−(max(min(max(−(*sl*,*ol*),−(*sl*,*ol*)),−(*ol*,3.397377)), *ol*),max(*ol*,min(×(−(*ol*,3.397377),*ol*),*ol*)))),ceil(−(−(*ol*, 3.397377),max(min(*sl*,−2.5250282),−(*sl*,+(max(−(*ol*, 3.397377),ceil(−(−(−(*ol*,3.397377),max(÷(sin(sin(*ol*)), −(*sl*,*ol*)),−(*sl*,*ol*))),max(−5.323125,−(*sl*,*ol*))))), ÷(+(min(max(×(−(*ol*,3.397377),*ol*),*ol*),max(*sl*, max(−5.323125,×(protectedlog$_{10}$(floor(7.62898)),*ol*)))), max(max(−(*ol*,3.397377),−(−(−(*ol*,3.397377), protectedlog$_{10}$(*ol*)),max(÷(×(−(*ol*,3.397377),*ol*),−(*sl*,*ol*)), −(*sl*,*ol*)))),sin(max(÷(floor(7.62898),−(*sl*,*ol*)), ÷(sin(−(*sl*,*ol*)),−(*sl*,*ol*)))))),ceil(*ol*))))))))
GE	min(pow(3.49,−(*ol*,−(*sl*,min(−(÷(−(6.45,*ol*),2.04)),*sl*))))),*ol*)

Table 5.2: Some of the optimal security policies inferred with GP and GE.

both distance and direction for Gaussian fuzzification algorithm.

For GE based experiments, we observe that the use of effective crossover and sensible initialisation provide very limited performance gain. In comparison to the results obtained using GP with similar parameter settings, the performance of GE is much worse. An analysis of the results reveals that the populations in many runs prematurely converge and get stuck at local optima at a very early stage. The two common local optima are the function that maps every possible input pair to *ol* and a constant function that maps every possible input pair to band 0.

5.3 Example Security Policies Inferred

The optimal policies evolved with the regression based approach using GP and GE are shown in Table 5.2. Each of these policies is the one with the best performance in all the 100 runs of each approach. The policies discovered using the rule based approach and ensemble approach are too complicated to be analysed manually. Other optimal policies inferred using either approach have very similar structure and size.

An immediate observation is that the size of the policies inferred using GE is much smaller compared to the ones inferred using GP in terms of the number of nodes. This can be explained by the fact that the evolution operators randomly select and change different parts of the individuals and producing individuals conforming to a grammar defined in GE with these operators is much more difficult than producing individuals conforming to the type correctness constraint imposed in GP. Additionally, the policies discovered with GP contain a deal of self-similarity. This suggests that some of these subcomponents have survived through multiple generations and crossover among the individuals.

In order to visualise the performance of these policies, we use a three dimensional plot where the x-axis and z-axis correspond to the input values of sl and ol, and the y-axis corresponds to the output values of the policies, i.e., the risk bands. Figures 5.5 and 5.6 show the output values of the target policy defined in (5.10) and these inferred policies respectively. It is interesting to note that the output values of the inferred policies change in a smoother fashion when the sl and ol values are high (especially the one inferred with GE). This seems to suggest that these inferred policies are more capable in assigning the appropriate risk band to a given access request than the manually designed target policy. This supports our initial claims that security policy inference from previous decision examples is in fact possible and these inferred policies can be a potential means to verify (if not refine) the currently implemented policy.

5.4 Evidence for the thesis and future work

Formulating an optimal security policy is difficult. Current research work attempts to alleviate this issue by looking for ways to analyse and refine security policies in a top-down manner. We propose an alternative view on this issue: inferring security policies from decision examples. This idea is entirely novel. There is no previous work to my knowledge in the application of EAs or machine learning techniques in inferring security policies.

In this chapter, we present some experiments that have been carried out to validate this proposal using EAs. Three different ways of representing security policies and the use of two different EAs are demonstrated. The results show that the inference process is largely independent of many parameters. We also show how the fuzzy set ensemble based approaches can be easily integrated into

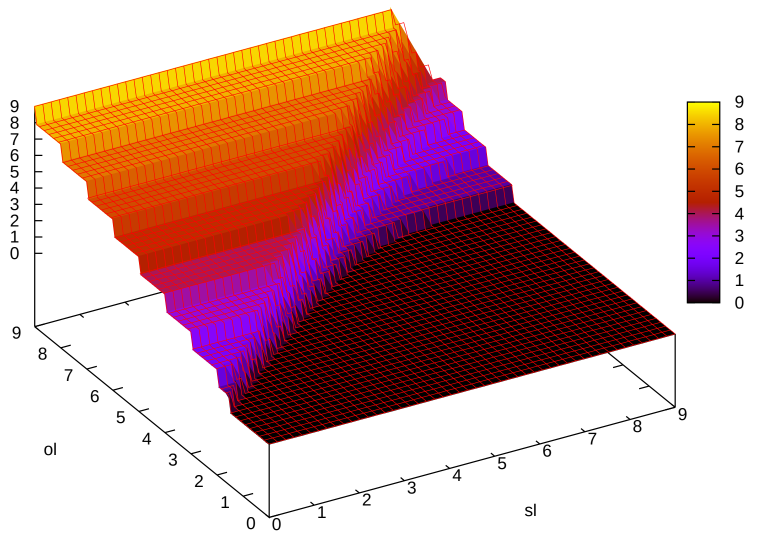


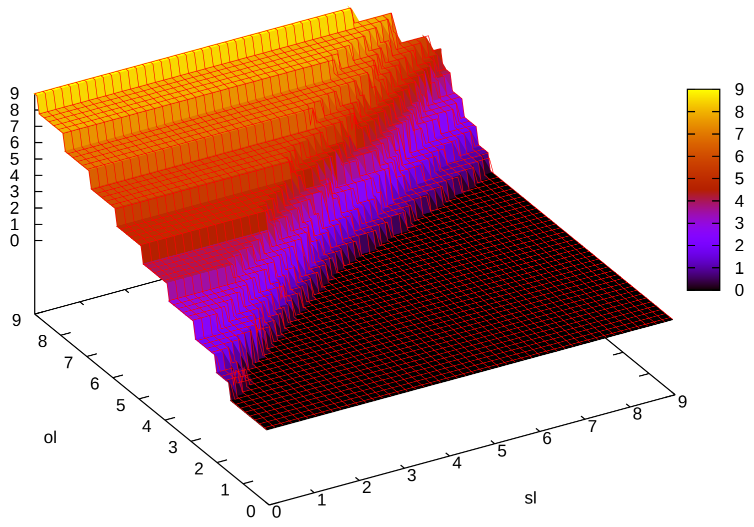
Figure 5.5: The output risk bands of the target reference model defined in (5.10).

the policy inference framework to enhance the inference ability, yet it remains an interesting research topic to search for the optimal ways of defining the underlying target fuzzy membership functions.

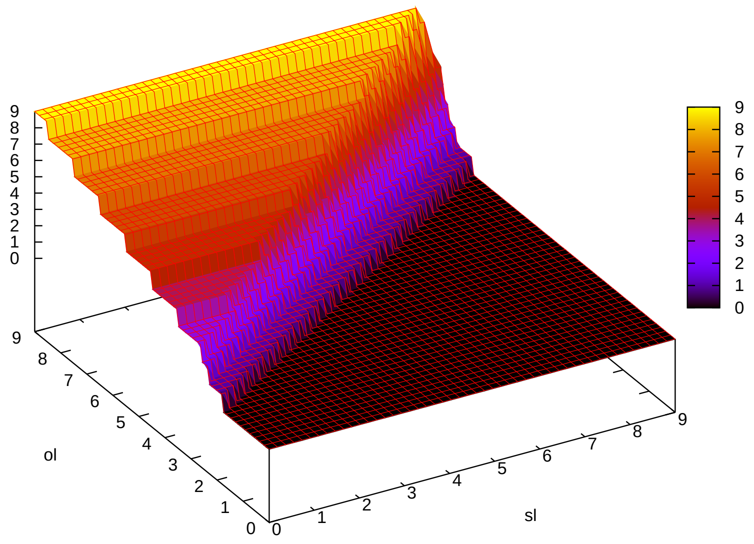
EAs have shown great potential in determining the security policies in the experiments. In particular, EAs are found to be able to quickly infer security policies with considerable complexity. The performance of these inferred policies is comparable to the original reference models that are used to generate the training sets. These techniques are also able to scale well with the range of input/output variables and to tolerate "wrong" examples in a training set. An obvious way forward is to validate this concept with other inference approaches and make a recommendation on which approach is better for what circumstance.

Being a data driven approach, the representativeness of the training set is crucial. Indeed, the experiments show that even the inference of the simple MLS Bell-LaPadula model may fail because of this. Inference summarises rather than speculates; the techniques do not know how to handle an unseen case.

As in other applications of EAs, the fitness function used is vital in guiding the search. A poor fitness function may result in policies that are suboptimal.



(a) GP.



(b) GE.

Figure 5.6: The output risk bands of the inferred policies shown in Table 5.2.

Interesting future work would be to examine how to design a fitness function in a principled manner that is suitable for cost sensitive learning, in which different types of prediction errors are not equally costly. This is likely to be appropriate for a security policy in which leaking of high sensitivity information is obviously far more severe than leaking of low sensitivity information.

5.5 Conclusions

This chapter presents some proof-of-concept experiments that have been carried out to validate our proposal: inferring security policies from decision examples using EAs. It first presents the experiments on inferring some simple binary decision policies and continues with the experiments on inferring the Fuzzy MLS model, which is a more complicated multi-decision policy model. In all cases, the results show that EAs are able to infer policies that can approximate (if not refine) the original reference models that are used to generate the training sets. The technique is also shown to be able to scale with the range of input/output variables and to tolerate "wrong" examples in the training set.

For a dynamic environment, the ability to infer policy from examples alone is not sufficient. The inferred and learnt policies will eventually become suboptimal over time as the operational requirements change. The policy needs to be updated continually to maintain its optimality. The next chapter demonstrates how multi-objective evolutionary algorithms (MOEAs) can be used to achieve this goal.

Chapter 6

Dynamic Policy Inference

Recent research [8] has suggested that traditional top-down security policy models are too rigid to cope with the changes that inevitably occur in dynamic operational environments. There is a need for greater flexibility in security policies to protect information appropriately and yet still satisfy operational needs. In the previous chapter, we have shown that security policies can be learnt from examples using EAs. Given a set of criteria of concern, one can apply these techniques to learn the policy that best fits the criteria. These criteria can be expressed in terms of high-level objectives, or characterised by the set of previously seen decision examples. Nevertheless, this is not sufficient for dynamic operational environments where the risk factors are constantly changing. The learnt policy will eventually become suboptimal over time as the operational requirements change. A new requirement is thus needed: the security policy has to be able to continually change and adapt to the operational needs to maintain its optimality or else it will inevitably be circumvented in creative ways.

This chapter details the experiments on dynamic security policy inference. As there is no dynamic security policy model available and therefore no decision example available for us to work with, we designed a dynamic security policy model. This model is used to generate time varying decision examples for training and evaluation purposes. Then, we present two MOEA based dynamic security policy learning frameworks. The first one is based on Fan's intuition [130] and the other one is Diversity via Opposite Objectives (DOO) that is able to maintain the diversity in the population during the optimisation process. This protects the population from premature convergence and allows the concept drift in the policy to be continually relearnt. The results show that these frameworks are very promising. Reasonably good approximators to the model are able to be inferred from the examples using these frameworks.

The rest of this chapter is organised as follows: Section 6.1 reviews some data stream classification algorithms that provided the inspiration for the development of our dynamic policy learning frameworks. Section 6.2 presents a dynamic risk-budget based policy that is used as the reference model throughout the chapter. Section 6.3 presents the experiment details and results on inferring the dynamic policy from decision examples. It also details the analysis of various ways to select the best solution from a set of candidate solutions. Section 6.4 summarises the main contributions and gives some possible future work. Finally, Section 6.5 concludes this chapter.

6.1 Data stream classification

We briefly review some data stream classification algorithms that provide the inspiration for the development of our dynamic policy learning framework. See [131, 132] for more details on data stream classification.

Data stream classification is the process of constructing classification models from continuous data records. Data stream classification poses two challenges to traditional data classification algorithms: continuous data flow and concept drift. Continuous data flow prohibits a classification algorithm from making multiple passes on the data set. Over time, the distribution of the data may change. This change is typically referred to as concept drift. The classification training algorithms must be able to cope with this.

To train the required classification model, it is often assumed that that some of the records in the data stream are labelled and therefore can be separated to form the training stream. Initially, research work has been focussed on revising the traditional algorithms to include fading effects for the older examples. A previously learnt classifier is required to undergo revision and to relearn the new concept constantly. Using the decision tree classifier as an example, the decision tree/subtree is pruned, regrown or discarded as necessary [133]. The resulting algorithms are often complicated. Making matters worse, as these algorithms typically discard old examples at a fixed rate, the learnt classifier is only supported by the latest data. This usually results in large prediction variances [130].

The ensemble of classifiers is another approach that has become very popular in data stream classification. This approach offers several advantages over single model classifiers. Firstly, it offers an efficient way to improve accuracy. Secondly, its parallel nature is easy to scale. Data in the stream are divided into chunks; a classifier ensemble (which itself can also be a set of classifier ensembles) is built from each data chunk. These classifier ensembles are combined using different weights to form the ultimate classifier. Some of the criteria used to determine the weights include the time (sliding window), *estimated* accuracy using the latest data chunk and variation in the class distribution.

In [130] Fan presents a simple example to illustrate that old data which are consistent with the new concept can help learning. Instead of throwing away all the old data, he proposed a framework that dynamically selects one of the following four classifiers as the final classifier:

1. The optimal classifier trained so far without using the latest chunk of data.
2. The classifier trained by updating the optimal classifier in 1 with the latest chunk of data.
3. The classifier trained only with the latest chunk of data.
4. The classifier trained from scratch with the latest chunk of data and some old data samples that are (assumed to be) consistent with the latest chunk of data.

The final classifier chosen is the one with the highest accuracy on the latest chunk of data using cross validation. This classifier is then set to be the optimal classifier trained so far (classifier in 1) in the next classifier selection process.

Fan's intuition is that no one knows if the latest data chunk on its own is sufficient to train a better classifier than the previous one learnt. Instead of statically defining how much old data is to be used, Fan's framework lets the data make the decision dynamically.

Dynamic security policy learning is very similar to data stream classification. Each possible decision in the policy can be viewed as a class; the learning objective is to search for the classifier that best agrees with the examples in the data stream in a timely manner. In both cases, the amount of data will inevitably increase over time; the algorithm must be able to learn incrementally and cope with changes. Yet there is still a small distinction in terms of the learning time requirement. In dynamic security policy learning, the learning time requirement is much relaxed. Acceptable time frames range from a few minutes to hours. This frees us from the one pass data set constraint in data stream mining, i.e., old data can be revisited if necessary.

The design of our dynamic security policy learning framework begins with this intuition and shows how MOGP can serve as an elegant framework for dynamic learning. A novel idea is then proposed to further reduce the error rate and response time to concept drift.

6.2 A dynamic security policy model

Since there is no known dynamic security policy model, we introduce a new time-varying, risk-budget based security policy model here. This model is used for generating the training decision examples and serves as the reference model against which the security policies learnt are evaluated.

In a system using a risk-budget based security policy, each user is given a number of risk tokens that represents how much risk the system is willing to take with that user. To access a piece of information, a user offers the number of risk tokens he is willing to spend from his budget to pay for the access. The system evaluates the risk incurred in granting the access and allows it only if the user's offer is greater than or equal to the risk. Unlike the models introduced previously, the risk evaluation of an access in this new dynamic model may vary with time, hence the name of the policy.

In order to evaluate risk for an access, the definition of risk in the Fuzzy MLS model in (3.1) is reused here and is restated as follows for ease of reference:

$$\text{risk} = (\text{value of damage}, V) \times (\text{probability of incurring the damage}, P)$$

The probability of incurring any damage, P, is the union of the following four *independent* probabilities:

1. P_{CH}: The probability that the communication channel between the user and the system is compromised.
2. P_{IS}: The probability that the information system is compromised.
3. P_{HU}: The probability that the human user is compromised for whatever reasons, e.g., being tempted, malicious, careless, etc.
4. P_{PH}: The probability that the physical security of the user or the system is compromised.

It should be noted that P_{CH}, P_{IS} and P_{HU} exclude the probability of physical compromises that are covered by P_{PH}. The independence assumption among these probabilities may not hold and result in P being over-estimated. This is fine from the security perspective, especially given the fact that all these probabilities are only estimates to begin with.

To estimate P_{CH}, we consider the security levels of communication channel, S_{CH}, may be either secure ($S_{CH} = 1$) or not ($S_{CH} = 0$). $P_{CH} = 0$ only if $S_{CH} = 1$ and $P_{CH} = 1$ otherwise.

To estimate P_{IS}, we use the five information system security rating levels, S_{IS}, as outlined in Trusted Computer System Evaluation Criteria (TCSEC) [22]. We assume that S_{IS} is an integer in the range $[0, 4]$; the higher S_{IS}, the more secure the system is. S_{IS} is mapped to P_{IS} using an inverse exponential function such that $P_{IS} = 1/\exp(S_{IS})$.

To estimate P_{HU}, we consider the sensitivity levels of the subject (user), sl, and the object (information), ol. The sensitivity level of a subject represents the level of trustworthiness of the subject whereas the sensitivity level of an object indicates the level of damage incurred if the object is lost or misused. To map these sensitivity levels to P_{HU}, the sigmoid function in (3.5) is reused and restated here as follows:

$$P_{HU} = \frac{1}{1 + \exp((-k) \times (TI(sl, ol) - n))}$$

where $TI(sl, ol)$ is the temptation index that indicates how much the subject with sensitivity level sl is tempted to leak information with sensitivity level ol. TI in (3.4) is reused and restated here as follows:

$$TI(sl, ol) = \frac{a^{-(sl-ol)}}{m - ol}$$

The intuition for P_{HU} and TI can be found in Section 3.4.1. In our experiments, the same settings in the experiments presented in the previous chapter are reused: sl and ol are integers in $[0, 9]$, $a = 10$, $k = 3$, $n = 4$ and $m = 11$.

To estimate P_{PH}, we assume there are 10 physical security rating levels, S_{PH}, which of each is represented by an integer in $[0, 9]$. Higher rating level indicates that better physical security protection mechanisms are in place. The mapping function from S_{PH} to P_{PH} is $P_{PH} = (9 - S_{PH})/9$.

To estimate V, an exponential function such that $V = a^{ol}$ used in the experiments presented in the previous chapter is reused here.

To introduce dynamic changes to the security policy, the risk calculated is multiplied by a safety margin factor, α, which has a value that varies over time in accordance with the changing environment. The *evaluated risk* for an access to a piece of information therefore becomes $\alpha \times P \times V$. The value of α is set to be a real value in the range $[1, 3)$. This setting allows the risk value to vary in a reasonably large range, so that the evaluation on the frameworks used may be done in a more rigorous manner. In practice, the changes in the policy are likely to happen in a much smaller and smoother fashion.

To generate a reasonable set of decision examples, we make the assumption that a user is rational in making each access request to information in the following ways. Firstly, a user is able to estimate the risk associated with the access to a certain degree of accuracy. Secondly, the user always attempts to minimise the number of risk tokens spent on the access without generating too many responses that result in a denial of access.

To model this, we assume each user always makes an offer of $(\beta_{min}+\gamma)\times P\times V$, where γ is a random variable with a beta distribution, B, which has a mean value of 0.5 and a variance value of 0.05, i.e., corresponding to $B(2, 2)$. A beta distribution is used here because it has a finite range between 0 and 1 and the parameters are chosen so that the distribution is symmetric. The user adjusts its β_{min} over time based on the allow/deny responses he receives by using a counter. The counter is incremented if an access request is granted or decremented otherwise. After every 5 decisions, the user increases β_{min} by 0.1 if the counter value is negative or decreases β_{min} by 0.1 otherwise. The counter is then reset to zero. The value of β_{min} is initialised to be 0.5 less than the initial value of α.

Using the settings described above, the security policy is defined as:

> A real-valued function $\mathit{SP}(\mathit{riskFactor})$ where $\mathit{riskFactor}$ is a tuple of $\langle S_{CH}, S_{IS}, \mathit{sl}, \mathit{ol}, S_{PH}\rangle$ with the interpretation that an information access request x is granted to a user y if and only if the offer of the user made for this access, $\mathit{offer}_y \geq \mathit{SP}(\mathit{riskFactor}_{x,y})$.

Examples of access control decisions are generated using this policy. Each example is a tuple of S_{CH}, S_{IS}, sl, ol, S_{PH}, *offer* and *decision*. A set of 10000 examples is generated, the value of α is changed randomly within its range after every 1000 examples. GP/MOGP is used to learn the underlying security policy model from these examples.

6.3 Experimentation

We observe that the specification of the reference security policy model presented in Section 6.2 is actually quite involved. Here we seek largely to approximate the model with GP/MOGP.

To do this, we view the security policy model as a function that maps six decision-making factors, S_{CH}, S_{IS}, sl, ol, S_{PH} and *offer* to a binary *decision*. Each individual tree in the population is used to represent a candidate function (policy). The terminal (leaf) nodes can be one of the decision-making factors or an ERC, which takes a real value in $[-10, 10]$. The non-terminal (non-leaf) nodes are mathematical functions. The functions chosen are $+$, $-$, $\times$, $\div$, protectedln(x)[1],

[1] $$\text{protectedln}(x) = \begin{cases} \ln(|x|) & \text{if } x \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$\exp(x)$, $\text{pow}(x, y)$, $\text{protectedlog}_x(y)$[1], $\max(x, y)$, $\min(x, y)$, $\sin(x)$ and $\cos(x)$.

In each experiment, a security policy is learnt and refined continually using a set of decision examples. Ideally, the learnt policy should generate the same decisions prescribed by all the training examples. In practice, the objective is to minimise the percentage of output decisions that are different from the ones generated by the true model. This percentage is the *error rate* of the learnt security policy. Here this error rate is estimated using the 1000 examples generated from the same model.

All the experiments described in this chapter are carried out using ECJ 18 [126] with the SPEA2 module obtained from ECJ 19[2]. The default parameter files in ECJ are used, i.e., `koza.params` for GP based experiments and `spea2.params` for MOGP based experiments, unless otherwise specified.

GP (and EAs in general) is stochastic in nature; the evolution process in each run may vary depending on the random seed used. To evaluate the performance, each experiment is repeated 100 times with a different random seed. The performance is evaluated by two criteria:

1. The median error rate of the best individuals in the 100 runs. This measurement indicates the quality of the security policy that can be learnt.

2. The number of the best individuals with error rates ≤ 0.25. This measurement indicates the likelihood of the learning in resulting a reasonably good security policy.

6.3.1 Static policy learning

To prepare for the experiments on dynamic policy learning, we started with three experiments to learn a static policy from decision examples (in a similar fashion to the experiments presented in the previous chapter). The target policy is, in a sense, static because all the training examples are generated from the reference policy model as presented in Section 6.2 with the same value of α.

[1] $\text{protectedlog}_x(y) = \begin{cases} \log_{(|x|)}(|y|) & \text{if } (x \neq 0 \text{ or } 1) \text{ or } (y \neq 0 \text{ or } 1) \\ 0 & \text{otherwise} \end{cases}$

[2] The reason of using ECJ 19 is that the SPEA2 module in this release has been revised heavily to remove some bugs and to be fully compliant with Zitzler's specification [93].

Each experiment runs for 200 generations and the training set consists of 1000 of such decision examples. Experiment 6.1 uses GP with the default genetic operators and parameters specified in ECJ. Experiments 6.2 and 6.3 use MOGP to address the problems encountered in Experiment 6.1.

Experiment 6.1: Single objective GP

This experiment uses GP with the default genetic operators and parameters specified in ECJ. The binary tournament selection scheme [81] is used to select the individuals from the current population to breed the next generation of individuals. This scheme holds several tournaments, with each tournament randomly choosing two individuals from the current population and then selecting the fitter one of the two. Each selected individual (policy) has a probability of 0.9 of being subjected to a crossover operation and a probability of 0.1 of being reproduced. The individual fitness is its error rate. Let r_i be the six decision-making factors, $decision_i$ be the decision encoded in example i and $SP(r_i)$ be the decision of an evaluated policy SP on r_i. Then, the fitness function of SP is:

$$f_{all}(SP) = \frac{1}{n} \sum_{i=1}^{n} (SP(r_i) \neq decision_i) \tag{6.1}$$

In this case, $n = 1000$ and the values 1 and 0 are used to represent *True* and *False* respectively. Therefore, this fitness function is essentially the fraction of decisions that the evaluated policy gets wrong.

The median error rate of the 100 best individuals is 0.3555 with a 95% confidence interval of $[0.3487, 0.3640]$. The best of the 100 best individuals has an error rate $= 0.107$. However, only 12 out of the 100 best individuals have error rates ≤ 0.25 and more than half of them have error rates > 0.35. This suggests that many runs get stuck at local optima.

Analysis of the structures of the individuals in the population reveals some common problems in GP. The average individual size (number of nodes) in the population grows quickly and becomes very large. This can be a phenomenon of bloat (uncontrolled growth of the average size of the individuals), overfitting problems or both. No attempt is made to distinguish them here as both make the learning process more difficult, use more memory and require more evaluation time.

Experiment 6.2: Bloat control with SPEA2

In this experiment, the SPEA2 bloat control method [134] is used to alleviate these problems. This method introduces a new objective: minimising the individual size (the number of nodes in the individual tree). Let size(SP) be the size of an individual SP. Then, the fitness function for this new objective is:

$$f_{size}(SP) = \begin{cases} \text{size}(SP)/512 & \text{if } 32 \leq \text{size}(SP) \leq 512 \\ 32/512 & \text{if } \text{size}(SP) < 32 \\ 1 & \text{otherwise} \end{cases} \tag{6.2}$$

In other words, individuals with less than 32 nodes have the same f_{size} value as an individual with 32 nodes and individuals with more than 512 nodes have the same value as an individual with 512 nodes. This is to avoid both over-simplified and over-complicated solutions.

SPEA2 is an elitist approach. An archive is maintained so that the non-dominated individuals of a generation can be preserved in the archive and passed on to the following generation. The reproduction operator is removed, i.e., the crossover operator is applied with probability of 1.

The experiment is carried out with only these changes. The results show that the bloat control method is effective. The average individual size, memory required and evaluation time taken are reduced significantly. However, the performance improvement in terms of error rate is marginal. The median error rate of the 100 best individuals is 0.3545 with its 95% confidence interval of $[0.3419, 0.3591]$. The best of the 100 best individuals has an error rate $= 0.126$ and there are 22 best individuals that have error rates ≤ 0.25.

Result analysis also reveals that the diversity among constants appearing in the individuals decreases with each new generation. This is expected as some form of convergence is necessary if a population is to produce a solution for the problem in question. It is often that the desired constants will not appear in the initial population that are randomly generated; but the evolutionary process is expected to synthesise the required constants by joining the existing ones through the operators. If the constants converge prematurely before finding the appropriate ones, the evolutionary process may get stuck at a local optima.

This lack of diversity among constants is the key to our problem. This is revealed by an analysis of the reference policy model presented in Section 6.2. This model grants an information access request if and only if:

$$\begin{aligned} & \mathit{offer} \geq SP(\mathit{riskFactor}) \\ = & \mathit{offer} \geq \alpha \times P \times V \\ = & \mathit{offer} \geq \alpha \times P \times a^{ol} \end{aligned} \tag{6.3}$$

As P is in $[0, 1]$ and α is in $[0, 3)$ and *offer* is programmed to track $SP(\mathit{riskFactor})$, the value of constant a which is raised exponentially dominates (6.3). The error rate of an individual will remain high even if it implements the same inequality except with a different value of a. As the diversity of the constants decreases over generations, the chance of finding the correct value of a becomes even smaller. To conclude, this reference policy model although appears to be simple, it is actually not that easy for GP to find it.

Moreover, the rate of convergence among constants in our experiments is further accelerated by the following factors:

- Small chance of using a constant as a leaf node — In ECJ [126], each ERC (constant) of a range is implemented as a terminal class and each variable is implemented as a terminal class. Each terminal class has an equal probability of being selected to be a leaf node. As our experiments involve six variable terminal classes and 1 ERC terminal class, the probability of an ERC being chosen as a leaf node is only 1/7. Moreover, the relatively large ERC range also exacerbated the problem in finding an appropriate constant.

- No mutation operator — The default configuration does not use mutation. Without mutation, there is no new constant introduced to the population. The diversity among constants in the population decreases with each new generation.

- The use of an archive in SPEA2 — SPEA2 restricts the binary tournament selection to the individuals in the archive and therefore only the constants that appear in these individuals have the opportunity to be passed on to the next generation.

Further analysis also reveals that a substantial portion of the individuals in the archive share the same or similar higher level structures of their trees (assuming the root of the tree is the highest node). The diversity among the individuals is lost. We think the primary cause of this problem is again the absence of mutation and the use of an archive in SPEA2.

Experiment 6.3: Enhanced constant evolution

To overcome these problems, this experiment is setup with the following changes. Firstly, six identical ERC classes are used to make the probabilities of choosing a variable and a constant equal. Secondly, a mutation-only setting is used instead of the typical high crossover and low mutation setting. There are four reasons to do so. Firstly, mutation introduces new genes and therefore diversifies the population. Secondly, much empirical evidence shows crossover provides no advantage over mutation in GP. Thirdly, this setting frees us from tuning the probability parameters of applying crossover and mutation. Finally, mutation provides a means to introduce new individuals by simply allowing mutation to happen at the root node of an individual. The probabilities of applying mutation at the root node, at a terminal node and at a non-terminal nodes are changed from 0, 0.1 and 0.9 (default) to 0.125, 0.125 and 0.75 respectively. This new setting has an effect in increasing the diversity in the population as well as focussing more on the search for the right constants.

The results improve significantly. The median error rate of the 100 best individuals is reduced to 0.2225 with a 95% confidence interval of $[0.1595, 0.2789]$. The best of the 100 best individuals has an error rate $= 0.099$ and the number of best individuals with error rate ≤ 0.25 is 54.

Figure 6.1 shows the distributions of the best individuals in all three experiments at different error rate intervals. The distribution of the best individuals in this experiment has shifted significantly to the left, i.e., situated at lower error rate intervals, in comparison to the ones in Experiments 6.1 and 6.2.

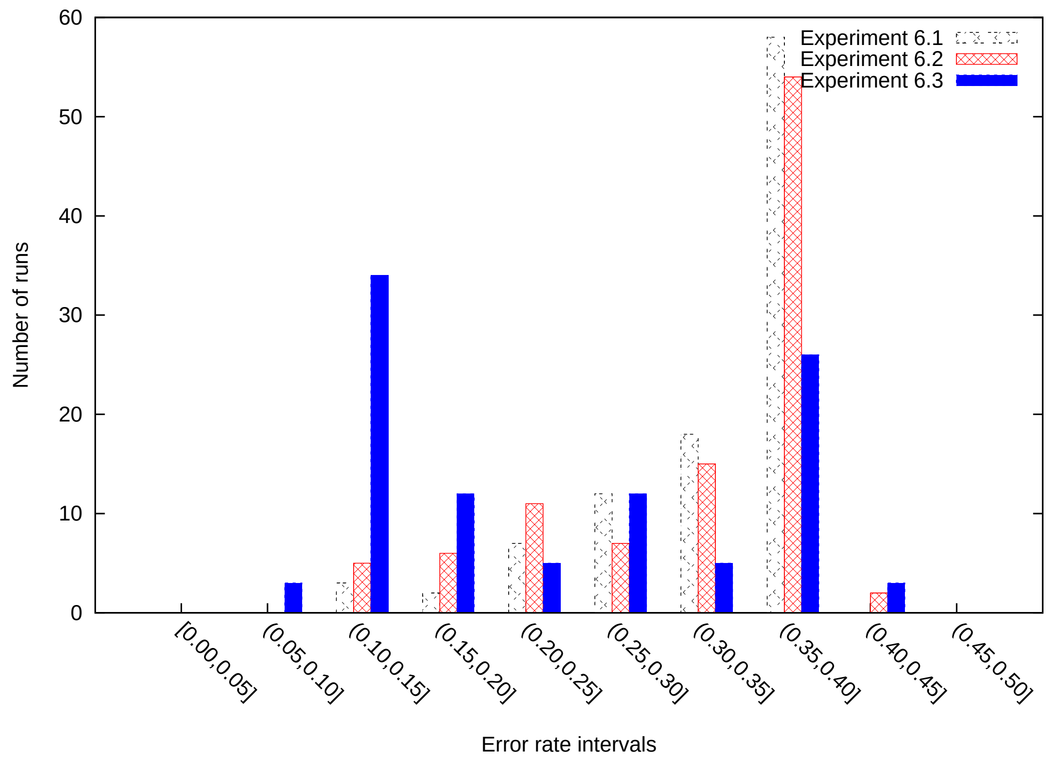
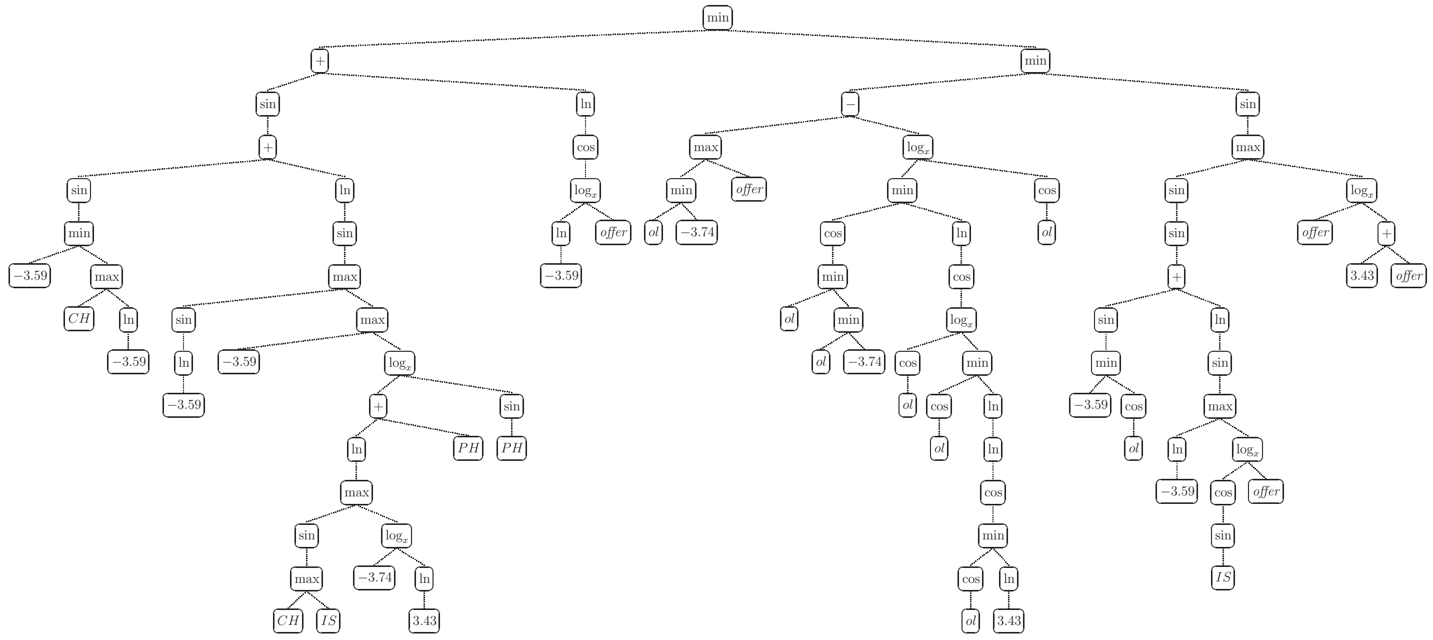


Figure 6.1: The distributions of the best individuals in all three experiments at different error rate intervals.

Example security policies inferred

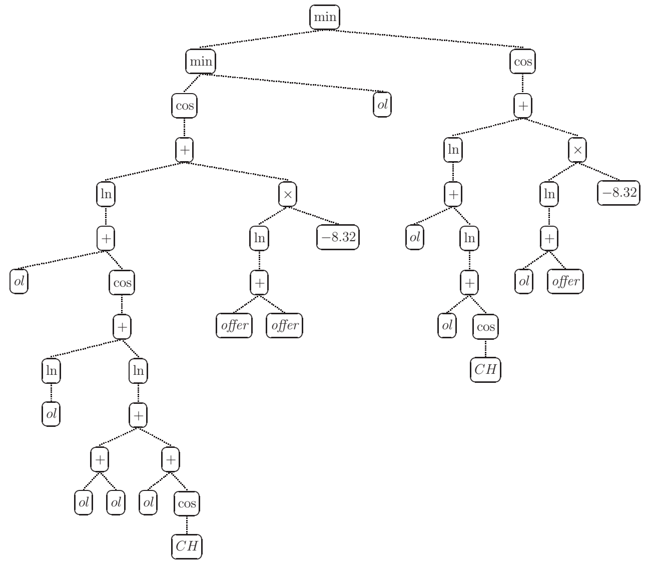
Some optimal policies inferred in Experiments 6.1, 6.2 and 6.3 are shown in Figures 6.2a, 6.2b and 6.2c respectively. Due to the space constraint, *type* is used to represent the factor S_{type}, $\ln(x)$ and $\log_x(y)$ operators are used to represent protectedln(x) and protectedlog$_x(y)$ operators, and all the constants are rounded to two decimal places. Other optimal policies inferred in these experiments also have very similar structure and size.

An immediate observation is that there is a huge reduction in the size and number of duplicate subcomponents in the policies inferred in Experiments 6.2

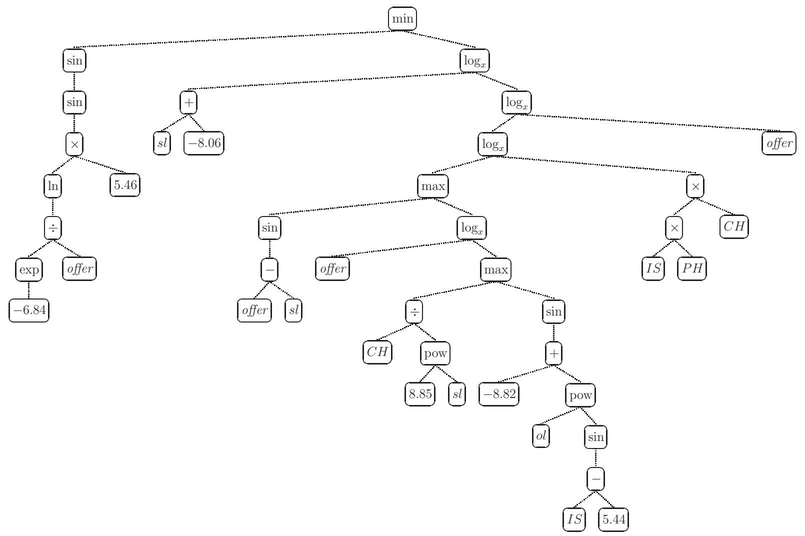

min
+ min
sin ln − sin
+ cos max log$_x$ max
sin ln log$_x$ min *offer* min cos sin log$_x$
min sin ln *offer* *ol* −3.74 cos ln *ol* sin *offer* +
−3.59 max max −3.59 min cos + 3.43 *offer*
CH ln sin max *ol* min log$_x$ sin ln
−3.59 ln −3.59 log$_x$ *ol* −3.74 cos min min sin
−3.59 + sin *ol* cos ln −3.59 cos max
ln *PH* *PH* *ol* ln *ol* ln log$_x$
max cos −3.59 cos *offer*
sin log$_x$ min sin
max −3.74 ln cos ln *IS*
CH *IS* 3.43 *ol* 3.43

(a) Experiment 6.1.

Figure 6.2: Some examples of the security policies learnt.



(b) Experiment 6.2.



(c) Experiment 6.3.

Figure 6.2: Some examples of the security policies learnt.

and 6.3. This serves as strong evidence that the use of SPEA2 to battle against bloat is effective.

Unlike the experiments presented in Chapter 5, it is impossible to show the mapping in these policies using a simple three dimensional plot as there are more than two risk factors. To answer the question if the target policy has been learnt in these experiments, the structure of the inferred policies is compared manually against the structure of the target policy. It is easy to observe that they are in fact not the same because some input factors are missing in the inferred policies. For example, sl is not found in the optimal policy inferred in Experiment 6.1, and sl, S_{IS} and S_{PH} are not found in the optimal policy inferred in Experiment 6.2. Nevertheless, with an error rate in the range $[0.099, 0.126]$, these inferred policies are likely to serve as good approximators of the target policy. This can be useful under certain operational environments such as MANETs, in which the evaluation of a complicated policy can be very expensive for a node.

6.3.2 Dynamic policy learning

So far, it is assumed that all the decision examples are available prior to the start of the learning process. This section investigates how to learn security policies dynamically when new decision examples become available gradually during the course of learning, some of which can be inconsistent with the ones that are previously seen. The learnt model is continuously refined, or even redefined if necessary, by these new examples.

Unlike before, these decision examples are organised into a sequence of data chunks. Each chunk consists of 200 examples to ensure that it has sufficiently good coverage and representation of the underlying policy. The policy changes every 5 chunks (1000 examples) by changing its value of α. In each experiment, the sequence is fed into the learning framework one chunk at a time. The first policy is learnt using the first chunk after 100 generations of evolution. Then, each subsequent chunk is used to refine the policy learnt from the previous chunks. Each refinement starts with the population of the last generation learnt from the previous chunks, and uses the examples in the latest chunk to learn a refined policy for a further 100 generations of evolution.

We first focus only on how to infer the optimal policy model from decision examples. At any time, the output model (from the learning algorithm used) is the individual (policy) with the lowest error rate on the latest chunk of examples in the evolving population. Three experiments are presented: Experiment 6.4 shows how dynamic learning can be performed by extending the current framework using Fan's intuition and Experiments 6.5 and 6.6 use Diversity via Opposite Objective (DOO) framework. We then show how ensemble approaches can be used to improve the optimal model selection problem at the end of this section.

Experiment 6.4: Fan's Intuition

In this experiment, the policy inference framework built in Experiment 6.3 is extended in two ways. Firstly, the number of examples, n, in the fitness function f_{all} is no longer fixed at 1000 but set to be the total number of examples received and therefore increases as more chunks are received. Secondly, a new fitness function is introduced to measure the error rate of a policy with respect to the latest chunk of data. Let r_i be the six decision-making factors, $decision_i$ be the decision in example i, s be the size of a data chunk and $SP(r_i)$ be the decision made by policy SP on r_i. Then, this new fitness function is:

$$f_{last}(SP) = \frac{1}{s} \sum_{i=n-s+1}^{n} (SP(r_i) \neq decision_i) \tag{6.4}$$

Along with this new fitness function, this experiment has three fitness functions in total, namely f_{all}, f_{last} and f_{size}.

The intuition employed here is similar to Fan's one (refer to Section 6.1). Each chunk of examples is used to refine the policy learnt from the previous chunks. The refinement process happens through 100 generations of evolution. In each generation, the policies with the lowest f_{all} or f_{last} values in the previous generation are preserved in the SPEA2 archive. These policies correspond to Fan's classifier 1. These policies are then refined with examples in the latest chunk through the evolutionary process. These refined policies correspond to Fan's classifier 2. New policies are generated (using mutation). Some may have the lowest f_{last} values within the population and correspond to Fan's classifier 3.

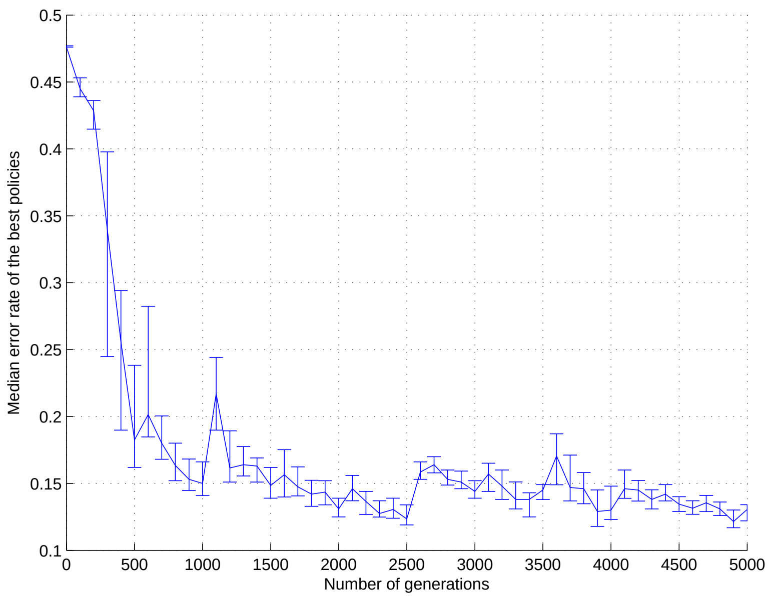
Others may have the lowest f_{all} values and correspond to Fan's classifier 4. After 100 generations, the policy with the lowest f_{last} value is chosen to be the new learnt policy.

The results are shown in two resolution levels. Figure 6.3a shows the median error rate of the best policies learnt after every 100 generations of training time. With the exception of the initial 500 generations, the median error rate is kept to be ≤ 0.220 at all times. For more details of the learning process, Figure 6.3b shows the median error rate of the best policies learnt in each generation.

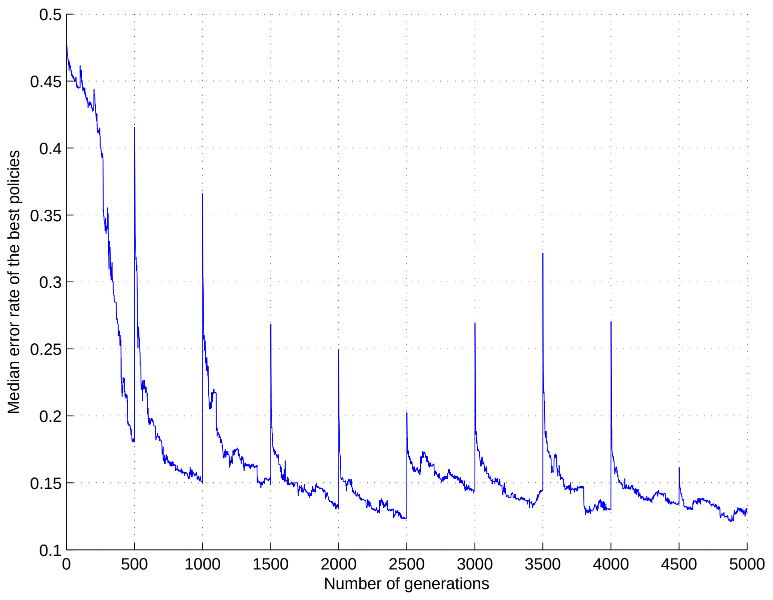
When a policy change happens every 5 chunks, i.e., $5 \times 100 = 500$ generations, the median error rate spikes up sharply. After the spike, the error rate decreases faster than in the initial 500 generations. This is a direct effect of using the two fitness functions: f_{all} and f_{last} together. This setting protects the optimal individuals with respect to the old policy from being eliminated too quickly and therefore allows these individuals to have chances to pass on the knowledge they have learnt to the new individuals. Consider the population in the generation prior to the change, the individuals that have low f_{last} values are also likely to have low f_{all} values. After the change, their f_{last} will become worse (higher), but their f_{all} values would only be affected slightly. Thus, these individuals will still have good chances of being kept in the archive and brought forward to the next generation.

We also observe that there is a strong correlation between the height of the spikes that happen every 500 generations and the change in value of α that is used to simulate policy change. The height of a spike is higher when the change in α is larger and vice versa. To see this, the values of α used to generate training examples and its changes, $\delta\alpha$, at every 500 generations are detailed in Table 6.1. An immediate observation is that the heights of the spikes produced at generation 3500 ($\delta\alpha = -1.6670$) and 4000 ($\delta\alpha = 1.6050$) are much higher compared to the spikes produced at generation 4500 ($\delta\alpha = -0.0587$). In practice, this may not be a major issue as the changes in security policy are often small and smooth.

Additionally, it is found that individuals in the archive converge and become more alike to one another over generations. However, this is not a problematic issue if the "structure" of the policy is learnt before the diversity is lost. The



(a) after every 100 generations.



(b) after each generation.

Figure 6.3: The median error rate of the best policies learnt in Experiment 6.4.

Generation	α	$\delta\alpha$
0	1.8648	1.8648
500	2.3314	0.4666
1000	1.4015	−0.9299
1500	2.2303	0.8288
2000	1.4048	−0.8255
2500	2.2718	0.8670
3000	2.8138	0.5420
3500	1.1468	−1.6670
4000	2.7518	1.6050
4500	2.6931	−0.0587

Table 6.1: The values of α and its changes, $\delta\alpha$, at every 500 generations.

subsequent changes require only changes in the value of α, which mutation can easily provide. Again, the loss of diversity also explains the very sharp upward spikes in the error rate during policy change as shown in Figure 6.3b. As the individuals are all alike, none of them matches the new policy well and thus the error rate increases sharply. However, the new policy is relatively easy to learn and thus the quick decrease in the error rate.

In summary, this approach can perform well under the assumption that the policy changes are relatively small and the knowledge learnt previously aids the learning process of the new policy. The use of two fitness functions: f_{all} and f_{last} allows the knowledge learnt to be passed on from generation to generation. However, this approach still suffers from the loss of diversity among the individuals in the population. This puts the general applicability of this approach in question.

Diversity via Opposite Objectives (DOO)

The loss of diversity problem is not uncommon in EAs. The "survival of the fittest" principle in EAs provides the fitter individuals with higher chances to survive and to pass on their genes on to the individuals in the next generations. Consequently, the individuals in the population will inevitably become more alike over generations. If significant diversity is lost prior to the optimal individual being found, the population is said to have converged prematurely to a local optimum. To prevent this, EAs use mutation operators to introduce new random genes and individuals. However, the chances that these new random genes and individuals can provide improvement over the current individuals are very small. Therefore, they are highly unlikely to be preserved. In other words, diversity is generated and then lost from one generation to the next.

To overcome this problem, several dynamic learning algorithms based on EAs have been proposed in the literature. Most, if not all of them *first* attempt to produce individuals that are optimised for the problem-related objectives and *then* attempt to maintain the diversity among individuals in the population as much as possible [135]. Their settings are often ad-hoc and the algorithms are often complicated. A new dynamic learning framework — Diversity via Opposite Objectives (DOO) — is proposed here. DOO takes the opposing perspective; it *first* attempts to maximise the diversity among individuals in the population through generations and *then* uses the ever increasing diversity to help in finding the optimised individuals.

In EAs, performing evolutionary operations on a single individual can be viewed as searching for more optimised individuals from the position of the individual in the solution space. Similarly, performing evolutionary operations on a diverse population of individuals can be viewed as a parallel search in many different parts of the solution space. This parallel search has a much better chance of finding more optimised individuals than a search starting from just one individual. The ever increasing diversity in DOO results in a domino effect such that not only is the search done in parallel, but the search space coverage increases as the diversity increases. Consequently, the chances of finding a more optimal individual become higher and higher as the evolution proceeds. This same effect cannot be achieved by conducting many parallel single-objective EA runs because each run is likely to be trapped in a local optima.

In DOO, each and every objective is changed into a pair of opposing objectives. For example, the objective of minimising error rate in our problem is changed to minimising error rate and minimising accuracy ($1 -$ error rate). DOO then optimises all objectives using MOEAs. The opposing objectives in DOO ensure that an individual who is weaker for one objective is fitter for the opposite objective. Therefore, *no individual is dominated by others*, i.e., all individuals are at the Pareto front. Each and every individual has a fair chance of passing on its genes to the next generation. This prevents the population from convergence, i.e., diversity is at least maintained. Since the true Pareto front is already found at the start of the evolution process, the only job left for MOEAs is to improve the spread of solutions on the Pareto front. As a result, the population of a generation will have a wider coverage of the solution space than the previous generations, i.e., diversity increases as MOEAs drive the evolution process.

To understand how DOO works via MOEAs, the concepts of the solution space, S and objective space, O introduced in Section 4.4 are used. Referring to Figure 4.3, diversity among individuals in a population is essentially a measure of how uniformly solution points are distributed in S. Whilst f does not maintain the distribution of solution points, it is often true in many problems that f is a continuous function between these two topological spaces, i.e., a set of points near a point $s' \in S$ is mapped to a set of points near the point $f(s') \in O$.

As f is not injective, the inverse is not necessarily true, e.g., two solutions can be very different yet both can solve the same problem equally well. However, following the continuity assumption on f, it is reasonable to assume that a set of points in O that is far apart corresponds to a set of points that is far apart in S. DOO makes use of this assumption and attempts to maximise the diversity of solution points in S via maximising the diversity of points in O using MOEAs.

In the SPEA2 implementation of MOEAs, if the number of non-dominated points exceeds the archive size, the point that has the shortest Euclidean distance to another individual in the objective space is dropped. If two solutions have the same distance to their nearest neighbours, the tie is broken by comparing their distances to their second nearest neighbours and so forth. This process is iterated until the non-dominated solutions can fit into the archive. Essentially, the goal is to fill the archive with non-dominated points as uniformly and widely distributed over O as possible. Let error rate and accuracy be the pair of objectives, a possible Pareto front and points that are to be kept in the archive are depicted in Figure 6.4. As the optimal point with respect to each objective is located at the corner of the Pareto front, i.e., they are furthest apart from other points, they are guaranteed to be preserved in each generation until a better one is found. At the same time, there is a higher likelihood for a better point to be found as the diversity increases.

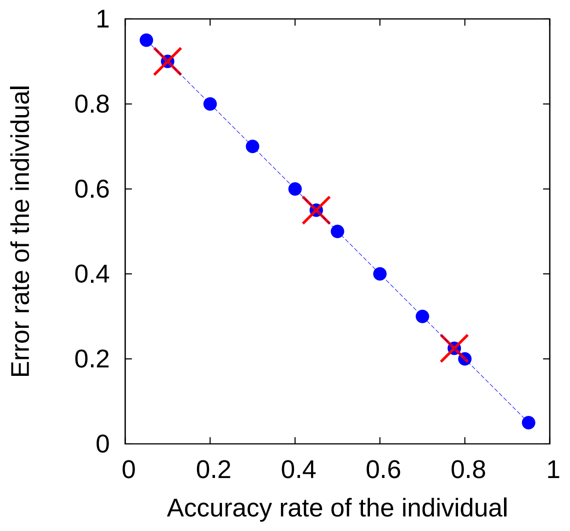


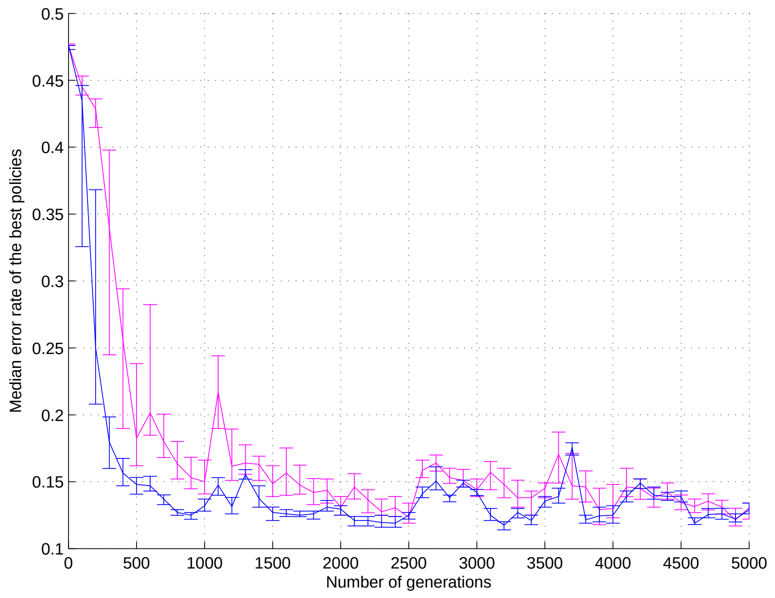
Figure 6.4: In the DOO setting, all individuals are at the Pareto front. SPEA2 removes individuals that are closest to others *iteratively* until they can fit into the archive. The individual at each corner is guaranteed to be in the archive.

Furthermore, all binary decision policies (all binary classifiers in general) can be inverted to their complements by a simple negation on their output decisions. Therefore, high error rate policies are just as good as those with a low error rate. To gain benefit from this, DOO selects the policy with the highest absolute value of bias which is defined as $0.5 -$ error rate. A negative bias value implies that the policy is optimised on the opposite objective and thus all its output decisions need to be negated if it is selected for use.

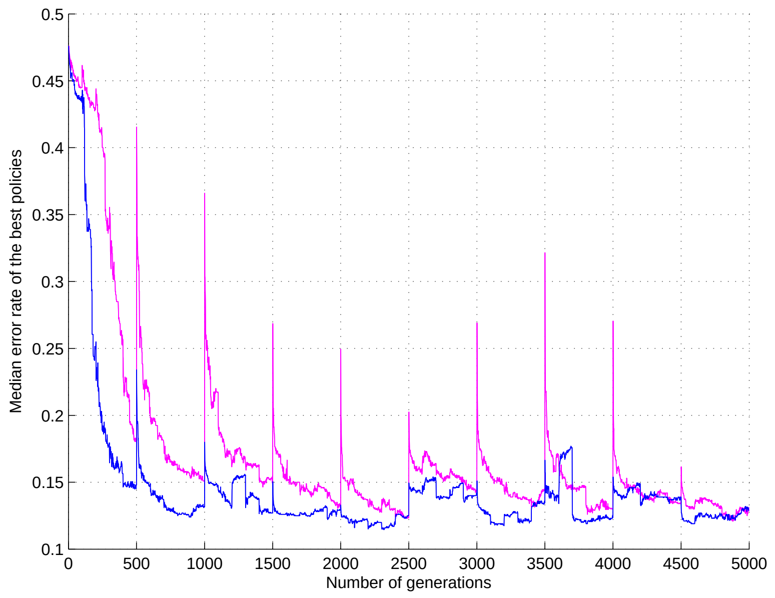
Experiment 6.5: Two pairs of opposing objectives

This experiment uses two pairs of opposing objectives: the first pair is f_{all} and $1 - f_{all}$, and the other pair is f_{last} and $1 - f_{last}$. We exclude f_{size} here as it is not a problem-related objective. All other settings remain the same as in Experiment 6.4.

Figure 6.5a shows the median error rate of the best policies learnt after every 100 generations of training time. Figure 6.5b shows the median error rate of the best policies learnt in each generation. The results of Experiment 6.4 that uses three objectives are also included in pink for comparison purposes.



(a) after every 100 generations.



(b) after each generation.

Figure 6.5: The median error rates of the best policies learnt in Experiment 6.4 (light pink) and Experiment 6.5 (dark blue).

In the initial 500 generations, the learning rate using DOO is significantly faster. The median error rate of the best policies is reduced to 0.250 in 200 generations. This result is comparable to the best static policy learning approach presented in Section 6.3.1, despite that the number of training examples used here being significantly less (400 vs. 1000) and these examples are being presented to the learning algorithm in sequential chunks.

Furthermore, the median error rate of the best policies learnt in each generation is lower than the one in Experiment 6.4. However, the error rates of the best policies may rise suddenly even in the absence of a policy change. Analysis of the results reveals that this is due to the way that the output policy is selected in each generation. Currently, the output policy in each generation is the one with lowest error rate/bias *estimated* using the latest 200 examples. As these 200 examples are generated randomly, they may not be sufficient to form a good representation of the target policy. Moreover, policy changes sometimes may simply mean revisiting a past policy. Therefore, the optimal policy in respect to the latest chunk may not be the true optimal policy. This effect is not obvious in Experiment 6.4 as all the policy candidates (individuals) in the population are very similar. In Section 6.3.2, we will show how ensemble approaches can be used to alleviate this problem. From the other perspective, this effect is a positive sign of diversity maintenance.

Lastly, the heights of the spikes in the error rate due to policy changes are much lower in DOO. Indeed, when a previously learnt policy is revisited, the error rate does not spike up at all. For example, the error rate does not spike up at all in generation 2000 as the target policy which has $\alpha = 1.4048$ is very similar to the target policy which has $\alpha = 1.4015$ learnt between generation 1000 and generation 1500. This is further evidence of diversity maintenance. A policy change can be viewed as a change in the fitness function, f. With a diverse set of policies maintained in the population, it is likely that one of them will be near the new target policy after the change.

Experiment 6.6: One pair of opposite objectives

An obvious weakness in the setups in Experiments 6.4 and 6.5 is that they are not very scalable. The evaluation of f_{all} involves scanning through all the decision examples seen. As the number of examples increases over time, the fitness evaluation time required for each individual increases. A possible way to overcome this is to only use a subset of decision examples, randomly sampled from all the decision examples seen. The likelihood of an example being sampled may also be set to decay over time, i.e., the older an example is, the less likely it is to be sampled.

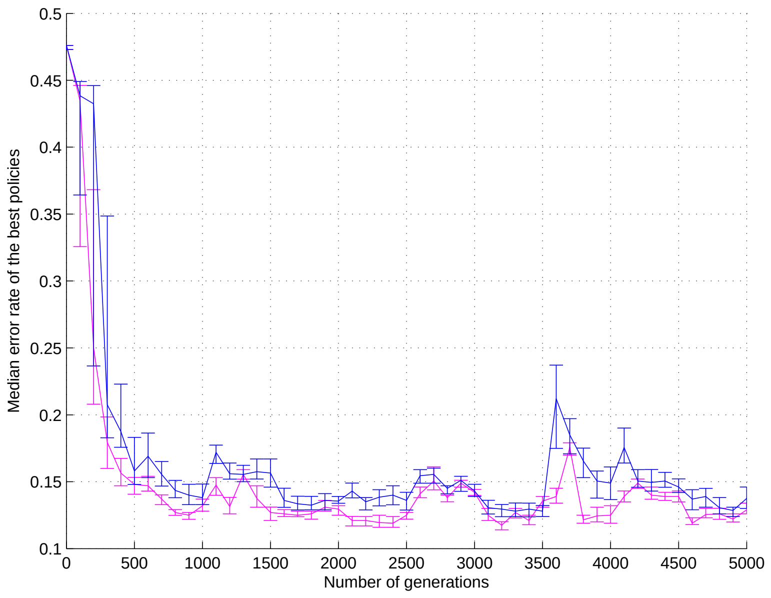
Prior to the search for a suitable sampling technique, it is always a good idea to check if the old examples are of any use. In this experiment, we drop the first pair of objectives from Experiment 6.5 to see the effect of not evaluating fitness against old decision examples. This experiment is carried out with only this change.

Figure 6.6a shows the median error rate of the best policies learnt after every 100 generations of training time and Figure 6.6b shows the median error rate of the best policies in each generation. The results of Experiment 6.5 are also included in pink for comparison purposes.

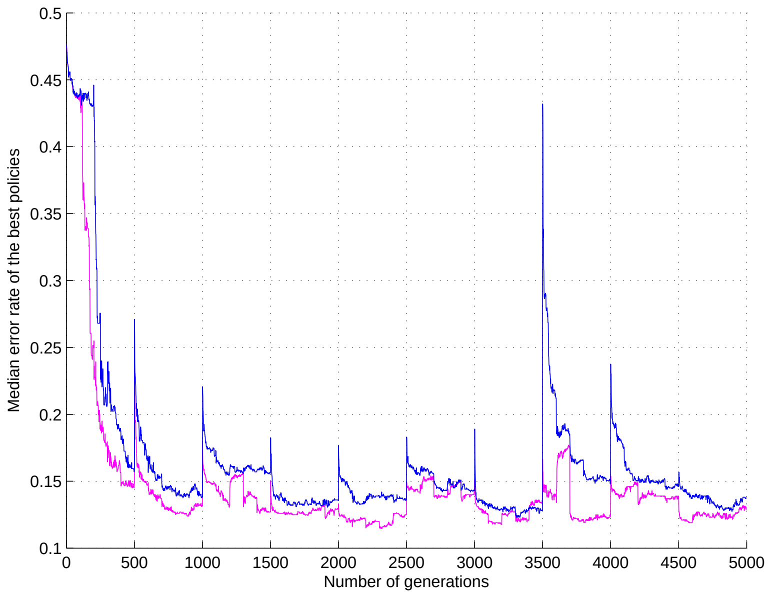
The results show that the performance of the best policies obtained in this experiment lie somewhere between those obtained in Experiment 6.4 and those obtained in Experiment 6.5. This suggests that the the old data (measured by the pair of objectives dropped) is indeed useful in maintaining a more diverse set of individuals in the evolving population.

Experiment 6.7: Selection of inferred policy models

This section examines various ensemble approaches to construct the ultimate model by combining multiple models to achieve better performance. We use a simple voting mechanism such that the output of the ultimate model is the majority output of all the models. This ensemble construction comes at virtually no cost in EAs; achieved simply by selecting the best n individuals from the final population.



(a) after every 100 generations.



(b) after each generation.

Figure 6.6: The median error rates of the best policies learnt in Experiment 6.5 (light pink) and Experiment 6.6 (dark blue).

However, the theoretical study of ensemble approaches has revealed two key factors that determine the performance of an ensemble: the performance of individual models and the diversity among all models in the ensemble [136]. As the population of an EA run with non-DOO setting converges and loses the diversity among individuals, the performance gain of using an ensemble is limited to the first factor. However, this is not a problem with DOO. Yet, we still have two questions to answer:

1. How many models should be used to construct the ensemble? Whilst the negative bias models are as useful as the positive bias models, the models with near zero bias are virtually useless. Should these models be included? If not, what should be the minimum threshold on the value of bias one model must have in order for it to be included?

2. Should the votes of all models have an equal weight? We choose to examine here if weighting the vote of each model with its bias using the latest chunk would provide better performance than a simple uniform weighted vote approach. If so, how much is the performance improvement?

We attempt to answer these questions by comparing the performance of ensembles built with the following combinations of models:

- Using the single highest bias model in the archive.
- Using the 8 highest bias models in the archive.
- Using the 16 highest bias models in the archive.
- Using the 32 highest bias models in the archive.
- Using the 64 highest bias models in the archive.
- Using all the models (128 models) in the archive.

The bias of each model is estimated using the latest chunk of decision examples.

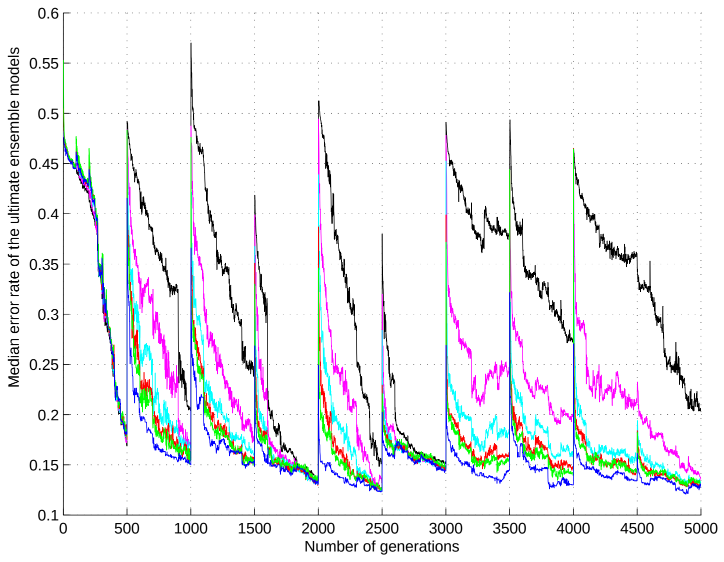
The models in each of these ensembles are combined with two different voting mechanisms: uniform weighted (unweighted) and bias weighted. In the bias weighted voting mechanism, the vote of each model is weighted with the absolute value of its bias on the latest data chunk. If the bias is negative, its vote goes to the complement decision class.

When the ensemble models are constructed from the models in Experiment 6.4, the error rates of the ensemble models do not decrease but increase with the number of models used as shown in Figure 6.7. This is because not all the models used for ensemble construction are optimised with respect to the error rate, some models in the archive are optimised with respect to other objectives, e.g., the model size. When the number of models used is small (8 or 16 models), it is very likely that the selected models are those optimised with respect to the error rate. However, as these models have converged to become very similar to one another, the formed ensembles do not result in any performance gain nor loss. As the number of models used increases, the ensembles begin to include those models that are not optimised with respect to the error rate. This causes the performance of the ensembles to become worse. The deterioration in performances is worse in the unweighted voting mechanism.

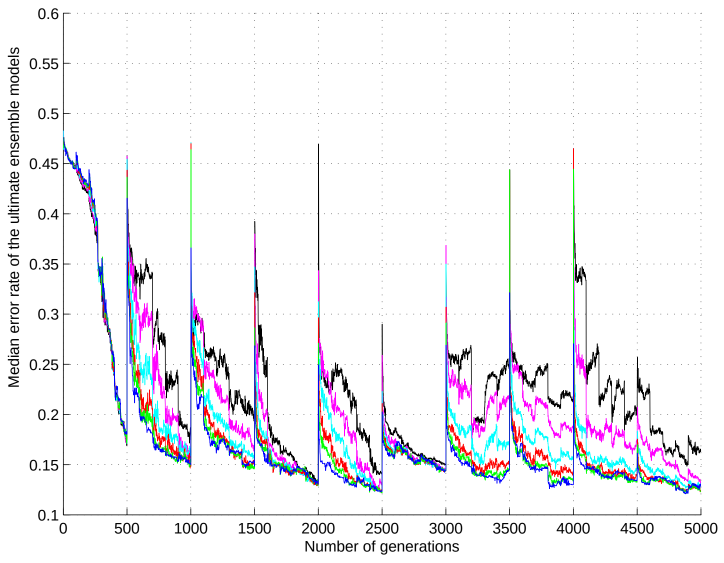
The performance of the ensembles formed using the models learnt in Experiments 6.5 and 6.6 are shown in Figures 6.8 and 6.9 respectively. When the number of models used is small (8 or 16 models), there is no significant change in performance. However, the changes in error rate become smaller; the sudden rises which tend to happen in the absence of policy changes seemingly disappear. This smoothing effect is especially clear between generations 2500 to 3000 and also between generations 3500 to 3700. As the number of models used increases, the performance becomes only slightly worse. This suggests that the diversity maintained in DOO provides a performance gain to counter the performance loss due to the use of lower bias models in an ensemble.

6.4 Evidence for the thesis and future work

This chapter details experiments that show how GP/MOGP can be used to learn as well as adapt security policy with changes. As there is no dynamic security policy model available to work with, a dynamic security policy model that varies over time is introduced here. This model is used to generate time varying decision examples for training and evaluation purposes.



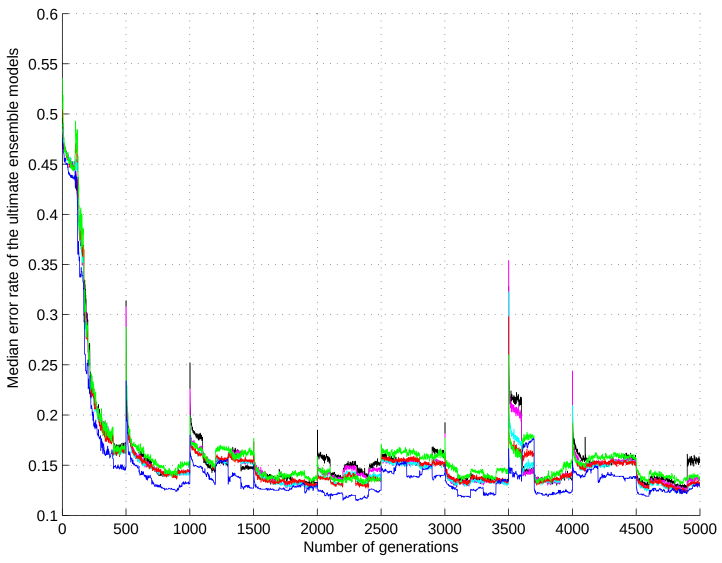
(a) Unweighted voting mechanism.



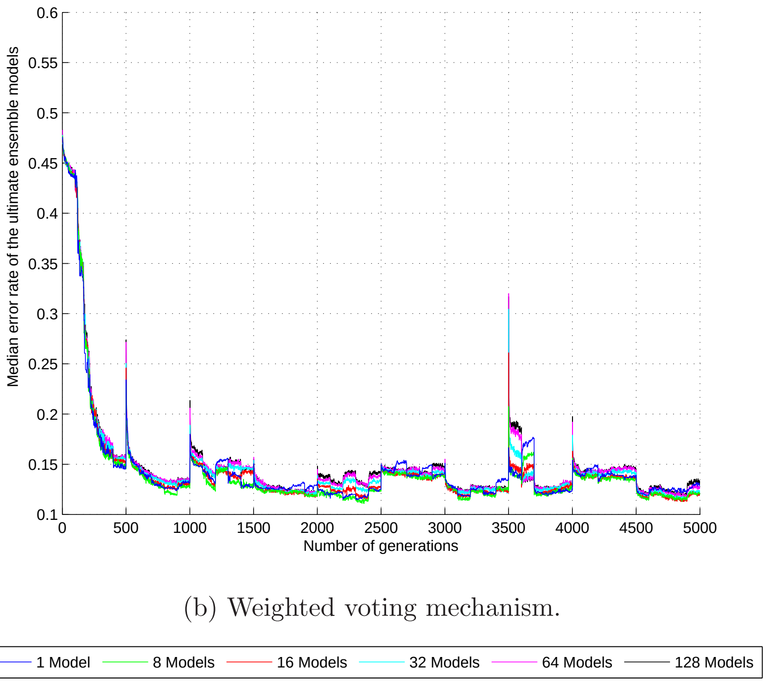
(b) Weighted voting mechanism.

1 Model — 8 Models — 16 Models — 32 Models — 64 Models — 128 Models

Figure 6.7: The median error rates of the ultimate ensemble models constructed from the models learnt in Experiment 6.4.

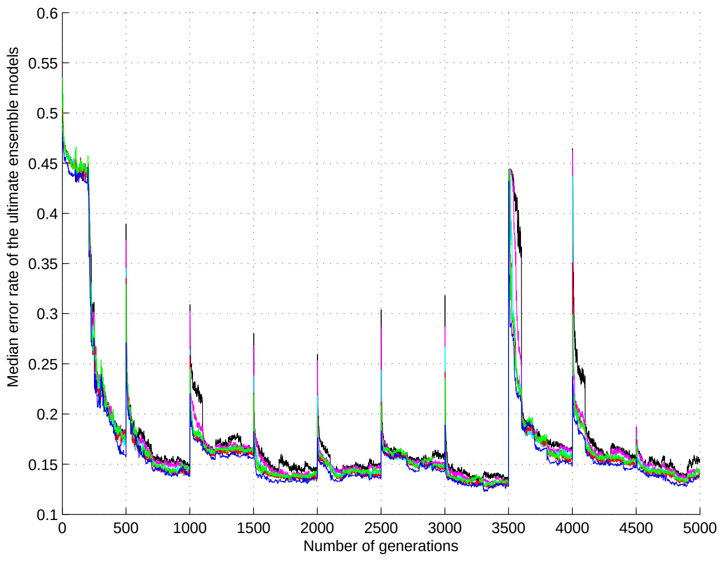


(a) Unweighted voting mechanism.

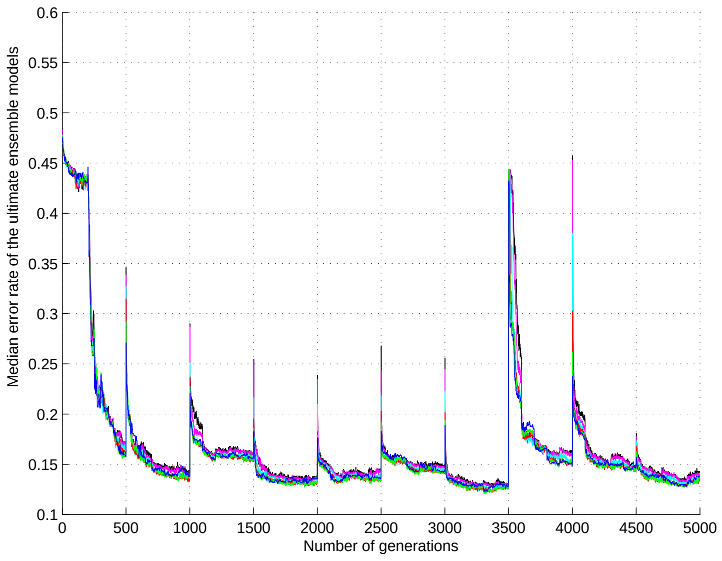


(b) Weighted voting mechanism.

Figure 6.8: The median error rates of the ultimate ensemble models constructed from the models learnt in Experiment 6.5.



(a) Unweighted voting mechanism.



(b) Weighted voting mechanism.

Figure 6.9: The median error rates of the ultimate ensemble models constructed from the models learnt in Experiment 6.6.

We first begin with three experiments on learning the designed security policy model statically as in the previous chapter. The results show some of the limitations with GP, in which the exact target policy can only be approximated at the best. This has a subtle implication on the applicability of our approaches. In practice, the security policies are likely to be much more complicated. It is unrealistic to expect the whole policy may be inferred from the decision examples using our approaches. A more realistic goal is to use our approaches in conjunction with the traditional top-down policy development approach. The high-level policies may first be refined into several subpolicies and these subpolicies are then used as the inference targets. Our inference approach can also be used as a security policy verification tool for the manually designed policies.

We also have found and proposed solutions to some common problems encountered when using GP, e.g., constant degeneration over the evolution process. As these are not the focus of this thesis, the solutions proposed here may be rather ad-hoc and specific to this problem. An obvious way forward is to examine these problems in detail and propose more rigorous solutions to them.

In the dynamic policy learning experiments, we proposed two novel dynamic learning frameworks based upon MOEA: one based on Fan's intuition [130] and DOO. In DOO, an n-objective optimisation problem is treated as a $2n$-objective optimisation problem, by adding an opposing objective to each of the original objectives. This allows the diversity in the population to be maintained whilst optimising the intended objectives. This diversity can aid in preventing the population from premature convergence and allows the concept drift in the policy to be continually relearnt. The results show that both frameworks are able to infer reasonably good approximators to the reference model from decision examples in an *incremental* manner. These frameworks also have better learning rate in comparison to the ones used for static learning. However, work remains to be done to make these frameworks more extensible. For example, searching for appropriate techniques to sample old decision examples.

Lastly, we also show how to build an ensemble policy model from multiple models in a single run to achieve better performance. In the DOO setting, this has the effect of reducing and smoothing the error rate of the inferred policies during concept drift.

6.5 Conclusions

We have argued earlier that some degree of adaptivity in policy is essential when the risk landscape is constantly changing. In some respects this applies to almost any system. However, the speed of evolution may vary very considerably. In the challenging dynamic environments referred to in the thesis hypothesis one might expect change to occur at a much faster pace than in fixed infrastructural networks. In this chapter we have investigated automated techniques for continually learning (and relearning) security policies from decision examples. Two dynamic security policy learning frameworks are proposed: one based on Fan's intuition and DOO. The results show that both frameworks are able to infer reasonably good approximators to the reference model from the decision examples in an *incremental* manner. Various ways of constructing ensemble policy model to achieve better performance are also examined. These approaches are found to be able to effectively reduce and smooth the error rate of the inferred policies during concept drift in the DOO setting.

Chapter 7

Mission-specific Policy Discovery

Recent research [8] has detailed why precanned one-size-fits-all security policies and mechanisms are too rigid for modern systems. In this chapter, we shift the emphasis away from specifying and refining a one-size-fits-all policy towards searching for a policy, from a family of policies, that has beneficial and acceptable outcomes. We believe this is entirely novel.

Here we will investigate if models of operational benefit and risk can be used to learn the optimal, or at least excellent, policies for specific scenarios. We use a risk-budget based policy family as an *example* only; it is a means to an end and stands as a proxy for any policy family from which we seek an instance best suited to the needs of a specific mission (or a specific family of missions). We employ the same techniques, namely GP/MOGP and DOO, to search over the space of policies, get feedback on the consequences of a particular policy, and home in on the optimal policies. These techniques have been shown to be effective in searching for optimal policies using decision examples in previous chapters. Other constraint solving or heuristic guided search approaches are potentially applicable.

The crux of the overall approach is that we need some notion of *feedback* to indicate how well a particular policy instance performs. Feedback can be obtained by labelled decision examples (used as training set in a similar way to the experiments in previous chapters), by static analysis, by numerical analysis (e.g., if we were to couch aspects of system behaviour as properties of Markov state transition graphs), or else by simulation. We choose to use simulation in this chapter to demonstrate the feasibility of our idea. Simulation is a highly flexible way to obtain feedback. It is of a particular use when the complexity of the system under examination prevents mathematical analysis.

The rest of the chapter is organised as follows: Section 7.1 introduces an operational scenario with a clear benefit and risk tradeoff in accessing information. Section 7.2 presents various proof-of-concept experiments to support our claims. Section 7.3 discusses the experimental results and Section 7.4 shows some interesting policies inferred in the experiments. Section 7.5 summarises the main contributions and points out some potential avenues for future research. Finally, Section 7.6 concludes this chapter.

7.1 Scenario: travelling across a battlefield

We present a highly simplified operational scenario where there is clear benefit from obtaining information and clear nasty consequences from allowing too much free access to it. The scenario is a battlefield with two teams of agents, blue and red. The battlefield is a 100×100 two dimensional grid. Blue agents aim to travel from an initial location, L_{init}, to a destination location, L_{des}, seeking to restrict casualties but also aiming for a quick traversal. Certain elements of the grid are in the hands of red forces and straying alone into an occupied grid position will lead to the liquidation of the agent and the compromise of all information it has had access to. At each time step, a blue agent can request further information about its vicinity, e.g., the location of other agents. It is assumed here that the amount of risk budget required for accessing the same piece of information under the same circumstance by a blue agent is the same. In other words, all blue agents are assumed to have the same levels of trustworthiness. Red agents attempt to prevent blue agents from achieving their objective by chasing and destroying any blue agent they see.

It is decidedly not the purpose of this task to determine how agents can best use the information they obtain. Rather, we assume there is *some chosen mechanism for using it, and the goal is to find the policy that then provides the best result.* It is possible that many of the same techniques we propose here could, mutatis mutandis, be used to search for an information use strategy, but we do not intend to address that issue here. We shall concentrate solely on the security policy instance discovery.

7.1.1 Movement strategy

At each time step, an agent can choose to move in any of the eight different directions to a neighbouring square: North (N), East (E), West (W), South (S), North East (NE), North West (NW), South East (SE) and South West (SW), or else remain at its current square (C). The decision-making process and movement of all agents are synchronous. At each time step, each agent decides where to move next based upon what he can perceive about the environment without knowing what decisions others make. Then, all agents move simultaneously based upon the decisions they have made. Two or more agents may end up occupying the same grid position.

What an agent can perceive is defined in terms of its sight distance, i.e., the distance an agent can see from its position. The notion of distance between two points is defined as the least number of steps an agent needs to move between the points. Initially, the sight distance is set to be one square for blue agents and two squares for red agents respectively. Blue agents may increase their sight distances to two squares by spending their risk budgets. Additionally, the grid map does not wrap around at its edges and thus the number of movement choices is more restrictive when an agent is at the edge of the map.

Local information of agents

Each agent is associated with several matrices in order to formally define its movement strategy. These matrices are the gradient distance matrix, the knowledge matrix and one or more direction selection matrices.

The gradient distance matrix, G, has the same size as the map, i.e., 100×100 two dimensional grid. It stores the distances of all squares in the map from the destination square. Each element $g_{x,y}$ in G represents the distance of the square with coordinate (x, y) from L_{des} on the grid map. This matrix is commonly shared by all agents.

The knowledge matrix, K, stores the knowledge a blue agent has acquired about the map so far. Each element $k_{x,y}$ in K represents the perceived risk associated with the square with the coordinate (x, y). At each time step t, a blue agent p updates its knowledge matrix, $K_{p,t}$ to account for the risk arising due to any red agent q that it can see from its current position. Formally, the update of $K_{p,t}$ can be written as:

$$K_{p,t} = \alpha K_{p,t-1} + (1 - \alpha)U_{p,t} \tag{7.1}$$

where α is the relative weight of the previous knowledge acquired and U is the update matrix. In this scenario, α is set to 0 for simplicity, i.e., blue agents have no memory about their past. Thus, (7.1) is simplified to $K_{p,t} = U_{p,t}$.

We have chosen a simple $U_{p,t}$ here such that each element $u_{x,y}$ in $U_{p,t}$ is as follows:

$$u_{x,y} = \sum_{i=1}^{\#q} \max(\text{sightRange}(i) - \text{distance}(x, y, x_i, y_i) + 1, 0) \tag{7.2}$$

where q is each red agent in the sight of the blue agent p and (x_i, y_i) is the location of the i-th red agent. In other words, a blue agent only considers the risk posed by all the red agents within its sight range. Each of these red agents poses a risk to any square within its sight range. The amount of risk posed to a square by a red agent depends on the distance between them. The further a square is from the red agent, the less risk is posed to that square. No risk is posed to any square outside its sight range. Figure 7.1 shows the risk posed by a red agent located at the centre of a 5×5 grid. As the sight range of a red agent in this scenario is fixed at 2, therefore the risk posed to the square where the red agent resides is 3, the immediate abutting squares (forming an annulus of squares) have risk that is equal to 2 and the next annulus of squares have risk that is equal to 1.

The direction selection matrices, DSM, is a 3×3 matrix. Each element in DSM represents the likelihood of a movement direction an agent chooses at each time step. For example, the central element, $d_{0,0}$ represents the likelihood of an agent choosing to remain at its current position whereas $d_{1,0}$ represents the likelihood of an agent choosing to move to the east. Figure 7.2 shows the

1	1	1	1	1
1	2	2	2	1
1	2	3	2	1
1	2	2	2	1
1	1	1	1	1

Figure 7.1: The risk posed to each square by a red agent located at the centre.

mapping between all the possible indices of *DSM* and the movement directions they represent. Additionally, the elements corresponding to movement directions that lead to invalid locations, i.e., locations that are not on the map, are always set to 0.

−1, 1	0, 1	1, 1
−1, 0	0, 0	1, 0
−1,−1	0,−1	1,−1

NW	N	NE
W	C	E
SW	S	SE

Figure 7.2: The mapping between the indices of a *DSM* and the movement directions they represent.

Movement strategy of red agents

As shown in Figure 7.3, the initial locations of all red agents are randomly distributed within a square-doughnut shaped patrol region defined in terms of three parameters: the centre, c, the inner radius r_{in} and the outer radius r_{out}. In this scenario, $c = L_{des}$, $r_{in} = 10$ and $r_{out} = 20$ are used.

Initially, each red agent randomly moves within the patrol region. When a red agent sees a blue agent, it begins to chase the blue agent. This pursuit ends after a finite number of time steps or as soon as the blue agent is killed. The red agent then returns again to the patrol region using the shortest possible path and starts randomly moving within the region as before.

More formally, a red agent can be in one of the following three states: Patrol (initial state), Chase or Return. The transition between states takes no time.

In the Patrol state, if a red agent does not see any blue agent, each element e_d in its *DSM* is set to $1/N$, in which d is a movement direction that will result in it remaining in the patrol region, and N is the total number of such directions. All

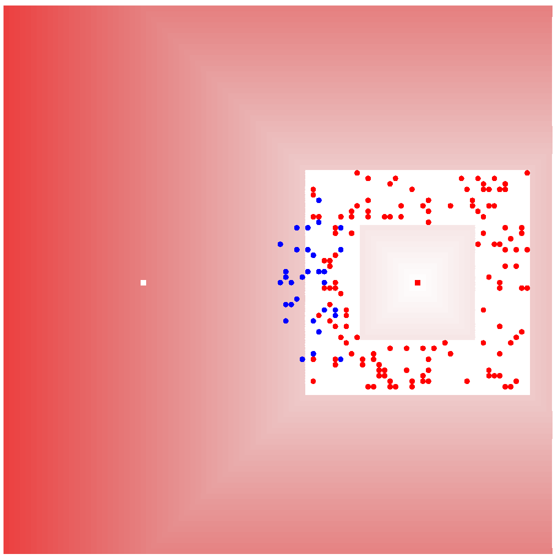


Figure 7.3: The simulated battlefield grid map.

other elements are set to 0. Otherwise, the red agent randomly sets its chasing target to one of the blue agents it sees, sets its chasing step to 10 and transits to the Chase state.

In the Chase state, a red agent has each element e_d in its DSM set to 1, d being the direction of the target blue agent. All other elements in its DSM are set to zero. Its chasing step is then decremented. It transits to the Return state if its chasing step runs out or its chasing target is killed or has escaped.

In the Return state, a red agent has each element e_d in its DSM set to 1, d being the direction of the centre of the patrol region as seen from its current location. All other elements are set to 0. In other words, an agent in this state will return to the patrol region using the shortest path and ignore any blue agent it sees on its way. Once a red agent arrives in the patrol region, it transits to the Patrol state.

Movement strategy of blue agents

The initial locations of all blue agents are randomly distributed within a distance between 0 and d_{start} from L_{init}. In this scenario, d_{start} is set to 7. At each time step, every blue agent attempts to move towards its destination gradually while seeking to avoid being killed by the red agents on the way. Once a blue agent arrives at the destination, the agent is considered to have achieved its goal and “disappears” from the map.

Formally, a blue agent can be in one of the following two states: Progress (initial state) and Arrive.

In the Progress state, a blue agent makes its decision as to where to move based upon the following factors:

1. The destination location, L_{des} — it should move towards its destination over time to accomplish its objective.
2. The location of red agents — it should try to move away from red agents to reduce the risk of being killed.

These factors are independently calculated using two DSMs, namely DSM_{des} and DSM_{risk}, and then aggregated using:

$$DSM_{final} = \beta DSM_{des} + (1 - \beta) DSM_{risk} \tag{7.3}$$

where β is the relative weight of each DSM. In our scenario, both DSMs have equal weight, i.e., $\beta = 0.5$. The direction that a blue agent will choose to move is the one with the highest element value in DSM_{final}. If there is more than one direction which has this value, a random choice is made among them.

To build DSM_{des}, we extract a 3×3 submatrix, G_{sub} from the gradient distance matrix, G. The submatrix to be extracted by an agent j is formed by the elements corresponding to the current square where j resides and its eight neighbouring squares. If any of these squares are off the map (i.e., the agent is at an edge of the map), the element is set to 0 and marked as invalid. For each of the valid elements, x, the following operations are performed:

1. calculateRelativeDifference(x) — set x to be the absolute difference of its current value from the largest value in G_{sub}. The closer a square is to L_{des}, the larger this value will be. The reachable squares furthest from L_{des} now have values of 0.
2. plusOne(x) — add 1 to x. This ensures that the values of all reachable squares are positive.

3. power(x, n) — raise x to the power of n. Larger values of n amplify the difference and thus increase the selection bias toward element x with higher value. Here, $n = 1$ is used, i.e., x is unchanged.

4. normalise(x) — x is normalised by dividing it by the sum of all the valid elements. The net effect of the above is to produce a probability distribution of moves to the valid squares; the closer square is to L_{des}, the greater the probability it is chosen.

DSM_{risk} is built in a similar way to DSM_{des} except that it uses knowledge matrix, K, as its source matrix (the matrix it extracts the element values from) and the parameter n in power(x, n) operation is set to 2.

An example of this scenario is depicted in Figure 7.3. The red agents randomly move in the patrol region (the white square-doughnut shaped region) with the destination (the red square in the middle of the region) as its centre. The blue agents are moving from their initial positions (region around the small white square at the left) to the destination. The intensity of the red background colour corresponds to the value in the gradient distance matrix.

7.1.2 Risk-budget based security policy

To integrate the risk-budget based policy into the scenario, a fixed amount of risk budget is assigned to each blue agent at the start. At any time step, a blue agent can opt to purchase extra information about its vicinity in the hope of reducing the risk of being killed. This leads to two questions: Firstly, what information is available for a blue agent to request? Secondly, when can a blue agent raise an information access request?

In the real world, information would be available at different levels of granularity. Higher granularity information costs more. Here the simulation is kept simple. Only one type of information is available to request: the locations of all agents at the distance of 2 squares from the current location of the agent. It follows that the cost of the information is not a matter anymore and therefore is simply set to 1 unit.

There is no straightforward answer to the second question. Instead of defining a fixed strategy regarding when blue agents can raise the information access requests, we decide to let each of them raise a request at every time step and it is the job of the policy to decide whether to grant or deny the request based upon the status of the agent. If the policy grants a request of an agent, the cost of the accessed information is charged against the budget of the agent, the information is annotated onto the knowledge matrix of the agent, and then the *DSM*s of the agent are recalculated. Otherwise, nothing happens.

To make an informed decision whether an information access request should be granted or not, the security policy needs to take into account the following risk factors:

1. The elapsed operation time, *currentTime*.
2. The price and granularity of the information requested. In this scenario, as there is only one type of information and therefore this factor can be omitted safely without loss of generality.
3. The remaining risk budget of the agent who raises the request, *budgetLeft*.
4. The current risk of the agent who raises the request, *currentRisk*. In this scenario, this factor depends solely on the number of enemy agents in its vicinity.
5. The estimated future risk of the agent who raises the request, *futureRisk*. In this scenario, this factor is estimated using the distance from its current location to the destination.

The security policy is thus defined here as:

> a real-valued function $SP(\mathit{riskFactor})$ where *riskFactor* is a tuple of $\langle \mathit{currentTime}, \mathit{budgetLeft}, \mathit{currentRisk}, \mathit{futureRisk} \rangle$ with the interpretation that an information access request is granted to an agent x if and only if $SP(\mathit{riskFactor}_x) \geq 0$ and $\mathit{budgetLeft}_x \geq 0$.

With the security policy integrated, the complete movement strategy of a blue agent in the Progress state is summarised in Algorithm 7.1.

```
foreach agent in BLUE team do
    calculate $DSM_{des}$ using gradient distance matrix
    calculate $DSM_{risk}$ using knowledge matrix
    calculate $DSM_{final}$ with (7.3)
    requestInformation(agent, policy)
    if request is granted then
        update $K$ with new information
        recalculate $DSM_{risk}$ using updated knowledge matrix
        recalculate $DSM_{final}$ with (7.3)
    move to the direction with the highest value element in $DSM_{final}$
```

Algorithm 7.1: The movement strategy of blue agents.

7.2 Experimentation

Given the scenario described above, we would like to search for an optimal instance in the policy space parametrised by these factors for blue agents. Optimality can have different meanings. In this context, an optimal policy is one that minimises the number of blue agent casualties as well and/or minimises the operation completion time (the time that the last agent is killed or arrives at the destination) given a fixed amount of risk budget. These two objectives can be in conflict with one another, e.g., minimising the operation time can be achieved by denying all information requests from the blue agents so as to increase their chance of being killed. GP/MOGP and DOO are used to search for the optimal policies here.

To search for the optimal policies using these approaches, each individual in the evolving population encodes a policy candidate. The terminal set, T consists of all the risk factors and a set of ERCs with values in the range $[-1, -1)$. The function set, F comprises $+$, $-$, $\times$, $\div$, protectedln(x), pow(x, y), exp(x), max(x, y), min(x, y), sin(x) and cos(x). To measure the fitness of a policy, the policy is executed by all blue agents in 10 scenarios described in Section 7.1. Each scenario is initialised with a different random seed and consists of 50 blue agents and 150 red agents. The mean of the policy performance in all the scenarios is used as the fitness value. The same set of scenarios is used repeatedly throughout the evolution process.

All the experiments described in this chapter are carried out using ECJ 18 [126] with the SPEA2 module obtained from the ECJ 19. The default parameters defined in `koza.params` and `spea2.params` that comes with ECJ are used, except for the following:

1. Number of generations: 100.
2. Population size: 256.
3. SPEA2 archive size (if applicable): 32.
4. Evolutionary operators (probability): crossover (0.9) and mutation (0.1).
5. Selection: binary tournament selection.

7.2.1 Experiment 7.1: Minimising casualty toll with f_{dead}

The objective of this experiment is to search for the optimal policy that minimises the casualty toll of the blue agents. The fitness function can be formed in a pretty straightforward way by setting it to be the normalised mean of the fraction of blue agents killed in a scenario. Formally, let N be the number of scenarios used, M be the initial number of blue agents in a scenario and c_i be the casualty toll of blue agent in scenario i. Then, the fitness of a policy, f_{dead}, is defined as:

$$f_{dead} = \frac{1}{NM} \sum_{i=1}^{N} c_i \tag{7.4}$$

7.2.2 Experiment 7.2: Minimising casualty toll with f_{msd}

Sometimes, the fitness function might not be so easily formed, perhaps because of cost or other difficulties in measuring it. In such cases, an indirect measurement can be used instead. In this experiment, an alternative fitness function, f_{msd}, which minimises the mean square distance between the final location of each agent and the destination, is used to achieve the same objective. Let N be the number of scenarios used, M be the initial number of blue agents and $f_{i,j}$ be the final location of blue agent j in scenario i (the location of a killed blue agent is the square where it was killed). Then, the fitness of a policy, f_{msd}, is defined as:

$$f_{msd} = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \left(\frac{\text{distance}(f_{i,j}, L_{des})}{\max(gridWidth, gridHeight)} \right)^2 \tag{7.5}$$

where the term "$\max(gridWidth, gridHeight)$" represents the greatest possible distance of any square from any possible L_{des}. Together with the term "$\frac{1}{NM}$", it ensures that the value of f_{msd} is between 0 and 1. For most choices of L_{des} (as long as L_{des} is not at the edge of the map), the achievable value of f_{msd} is strictly less than 1.

7.2.3 Experiment 7.3: Minimising casualty toll with DOO

To further investigate the applicability and generality of the DOO framework introduced in Section 6.3.2, we reuse it here to search for the optimal policy that minimises the casualty toll of the blue agents. To search with DOO, we need to add an opposing objective to each of the original objectives. We reuse f_{dead} defined in Experiment 7.1 here and define its "opposite" fitness function to be the normalised mean of the fraction of blue agents who remain alive in a scenario. Formally, let N be the number of scenarios used, M be the initial number of blue agents in a scenario and a_i be the blue agent that remains alive in the scenario i. Then, this "opposite" fitness function, f_{alive}, is defined as:

$$\begin{aligned} f_{alive} &= \frac{1}{NM} \sum_{i=1}^{N} a_i \\ &= 1 - f_{dead} \end{aligned} \tag{7.6}$$

7.2.4 Experiment 7.4: Multi-objective optimisation

In this experiment, the objective is to search for the set of policies that is optimal in two criteria: minimal casualty toll of blue agents and minimal mission completion time. The search is carried out using MOGP with SPEA2 implementation.

To use MOGP, the fitness function of each objective must be defined. To measure casualty toll, f_{dead} defined previously is used. To measure the mission completion time, f_{time} is defined as the normalised mean of completion time of a run. Formally, let N be the number of scenario used, T be the maximum completion time of a run and t_i be the completion time of scenario i. Then, the fitness of a policy, f_{time}, is defined as follows:

$$f_{time} = \frac{1}{NT}\sum_{i=1}^{N} t_i \tag{7.7}$$

Here T is set to be 100.

7.3 Experimental results and evaluation

To evaluate the performance of the optimised policies found, three baseline policy models, *NoAccess*, *FullAccess* and *RandomAccess* are created. *NoAccess* and *FullAccess* models are simple: they deny and grant, respectively, all access requests. Assuming that having more information will always help an agent in making a better informed decision in choosing the movement direction, these models essentially define the lower and upper bounds of the achievable performance. However, as will be shown later, the experimental results suggest that this assumption is not true; agents can use the new information in a way that can lead to a deterioration in their performance. In *RandomAccess* model, the decision to grant an access request depends on the availability of the current remaining budget held by the agent who raises the request. If the agent has sufficient budget to pay for the access, the access request is granted or denied with equal probability. Otherwise, the access request is simply denied. This model serves as the baseline model for a given initial budget of each agent.

In each experiment, four different initial risk budget allocation settings are attempted. 10, 20, 40 and ∞ units of risk are allocated to each blue agent for each mission. Each setting is repeated 10 times using a different random seed and the performance of each optimised policy is summarised in Table 7.1. In the MOGP experiment, the results shown are the performances of the optimal policies with minimal casualty toll of blue agents without taking into account the mission completion time. The median is used here instead of the mean as the distribution of the performance metric used in each experiment is unknown. In all cases, the results suggest that a policy can be optimised considerably for the 10 specified missions.

The performance of the optimal policies found in Experiments 7.1 (using f_{dead}), 7.3 (using DOO) and 7.4 (using MOGP) are significantly better than the unoptimised *RandomAccess* policies. This result serves as a strong evidence to support the idea of searching for the optimal policies for a set of missions using EAs. Additionally, the performance difference among these optimal policies is very small such that it is statistically insignificant. In other words, the policies optimised with DOO are as good as the ones optimised with the traditional GP/MOGP.

Optimisation in Experiment 7.2 (using f_{msd}) has a relatively small improvement when the risk budgets available are limited. Further investigation reveals that this is due to the bias in the fitness function used. The search using f_{msd} gives preference to a policy that leads each agent to be as close as possible to the destination *without* considering if the agent ever reaches its destination. This bias effect is more obvious when the initial risk budget is low and insufficient. As the initial budget increases, the performance difference between the policies learnt in this experiment and those learnt in other experiments becomes less. Indeed, in the case we set the initial budget to be ∞, the difference is very small and is statistically insignificant.

The results also contradict our initial intuitive assumption that using extra information is always beneficial. For example, the *RandomAccess* policy performs worse than the *NoAccess* policy when the initial budget of each blue agent is set to 10 or 20. Moreover, the optimised policies in all the experiments outperform the *FullAccess* policy when the initial budgets allocated to each blue agent is set to 40. In other words, the agent has to use the information at the right time to gain benefit from it. This reinforces our claims that specifying policy is a very difficult and counterintuitive task, and obvious solutions might be far from being optimal.

In the multi-objective optimisation experiment (Experiment 7.4), the output of each run is a set of non-dominated policies that attempt to approximate the real Pareto optimal set of policies. The result from one experimental run of each initial budget setting is shown in Figure 7.4. In each graph, the leftmost point

Budget	Median blue agent casualty tolls with 95% confidence intervals / %				
	Random	f_{dead}	f_{msd}	DOO	MOGP
NoAccess	$55.6(55.6, 55.6)$	-	-	-	-
10	$59.7(58.0, 60.6)$	$42.0(41.0, 43.4)$	$47.5(44.0, 48.6)$	$42.5(41.0, 43.8)$	$42.5(41.6, 44.2)$
20	$57.7(56.6, 58.6)$	$33.5(31.0, 35.2)$	$40.0(38.2, 42.8)$	$35.0(34.2, 36.2)$	$35.9(34.0, 37.2)$
40	$50.7(49.8, 53.6)$	$23.3(21.8, 26.0)$	$27.3(26.0, 28.6)$	$25.2(23.6, 26.0)$	$25.0(24.0, 25.8)$
∞	$51.4(50.2, 52.4)$	$23.7(22.8, 24.8)$	$24.1(22.0, 28.2)$	$22.9(22.0, 23.4)$	$23.6(23.0, 23.8)$
FullAccess	$31.6(31.6, 31.6)$	-	-	-	-

Table 7.1: The experimental result summary on the performances of the optimal policies with respect to the casualty toll of blue agents found using GP/MOGP and DOO.

corresponds to the policy instance that results in the lowest blue agent casualties while the lowest point corresponds to the policy instance that results in the minimal mission completion time. The points in between represent all other non-dominated policy instances. All other experimental runs produce similar results.

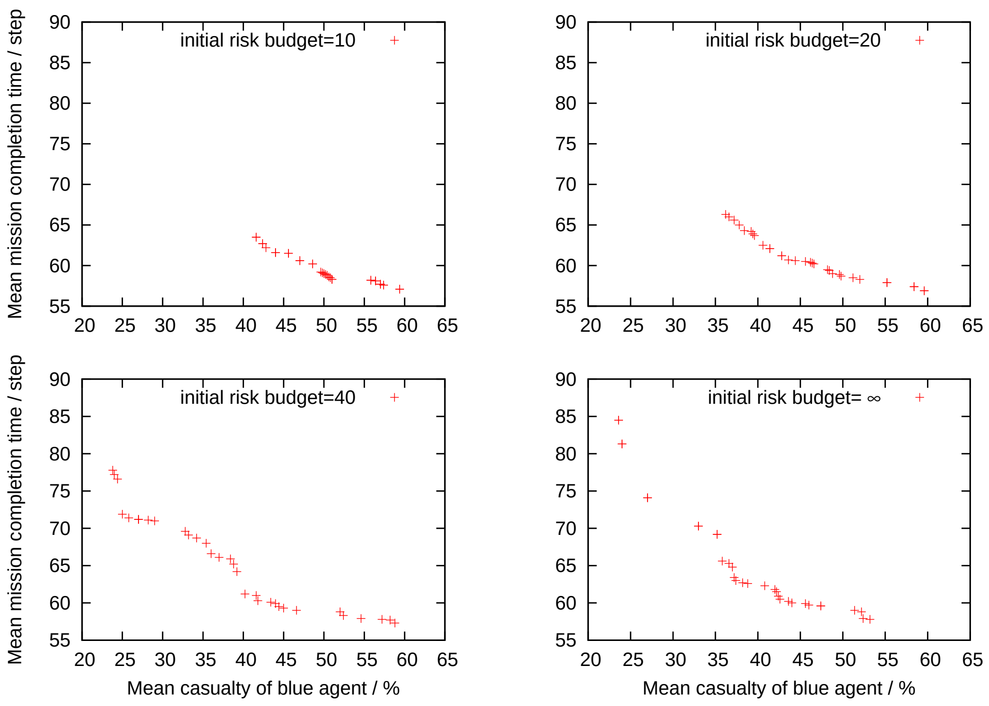


Figure 7.4: The non-dominated solutions in one run using different initial budgets.

The results of 10 runs for all initial budget settings are shown in Figure 7.5. The non-dominated policies found in one run can be dominated by the non-dominated policies found in other runs as EAs are stochastic in nature. the resulting policies depend on the random seeds used. However, policies found in all runs are not very far away from the Pareto front approximation formed by the non-dominated policies of all runs. This suggests that this approximation is very likely to be the real achievable Pareto front.

By observing the Pareto fronts of the solutions formed, one can immediately see that the increase in the amount of initial budget may be able to reduce the blue agent casualties but not the mission completion time. This information can

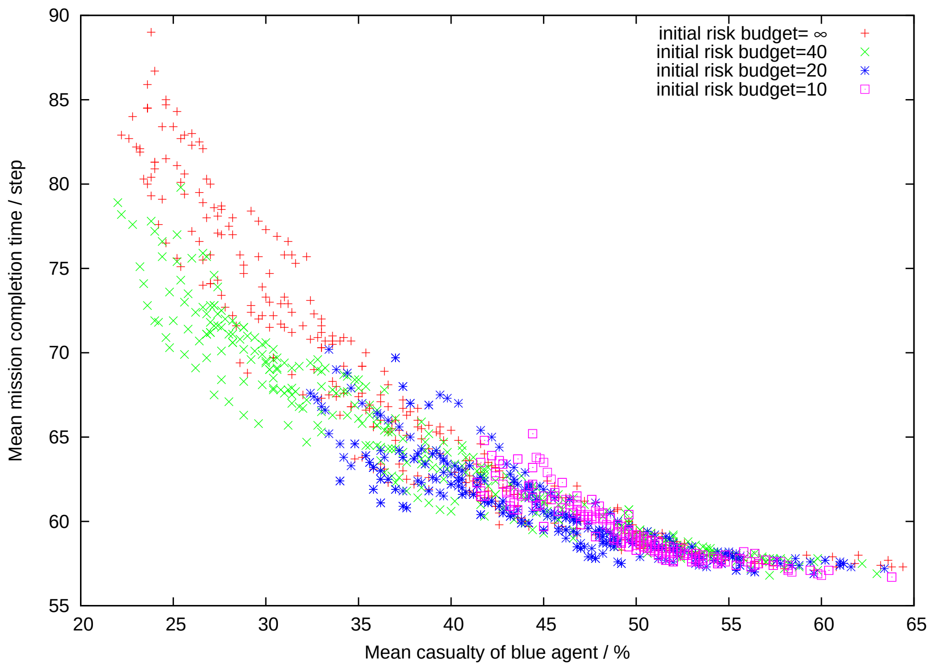


Figure 7.5: The Pareto optimal solution set.

be very useful in practice. For example, one might wish to have a policy that can complete the mission within 50 time steps. Based on the results, this is unachievable no matter what initial budget is given. On the other hand, given a fixed initial budget and an acceptable mission completion time, one could select the optimal policy from the front that results in the minimal casualty toll of the blue agents.

Additionally, the results suggest that the setting of 40 units of risk per blue agent for each mission is sufficient to accomplish 10 missions optimally with respect to the casualty toll of blue agents; extra budget does not help. This provides a way to estimate the optimal initial risk budget allocation. Alternatively, one can search for the minimal risk budget required to achieve the same optimal performance by simply setting this as an additional objective.

The results in the multi-objective optimisation experiment are also compared to those obtained in single objective experiments. This is done by comparing the median performances of the optimal policies with respect to the casualty toll of blue agents in all initial budget settings. The results show that the optimal policies found in both approaches have similar performance.

7.4 Example security policies inferred

Some examples of the Pareto optimal policies inferred in the multi-objective optimisation experiment with 40 units of risk budget per agent are shown in Figures 7.6, 7.7 and 7.8. Because of space constraint, the individuals are shown such that B, F, R, T and $\ln(x)$ operator are used to denote *budgetLeft*, *futureRisk*, *currentRisk*, *currentTime* and protectedln(x) operator, and all the ERCs are rounded to two decimal places. The policy shown in Figure 7.6 is one of the optimal policy with respect to the casualty toll of blue agents (one of the leftmost green crosses at the Pareto front in Figure 7.5), the policy shown in Figure 7.7 is the optimal policy with respect to the mission completion time (the lowest green cross at the Pareto front in Figure 7.5) and the policy shown in Figure 7.8 is the optimal policy with respect to the casualty toll of blue agents given the constraint that the mean of mission completion time has to be within 70 time steps.

These Pareto optimal policies discovered appear to be large and complicated. (The policies shown here are relatively simple in comparison to the others which are not shown.) It is also interesting to note the extensive use of min, max, sin and cos operators, which do not commonly appear in human designed functions. These findings concur with our intuition that the policy that is best suited an operational environment can be complicated and the task of manually specifying such a policy can be very difficult.

7.5 Evidence for the thesis and future work

This chapter introduces the idea of moving away from specifying and refining a one-size-fits-all policy towards searching for a policy that has beneficial and acceptable outcomes from a family of policies. It presents a simple scenario, in which evolutionary algorithms are used to discover the (near) optimal policies that fit the scenario. Here, GP/MOGP and DOO are used as optimisation tools to synthesise the optimal policies, in terms of achieving the mission as well as security objectives without violating some predefined constraints.

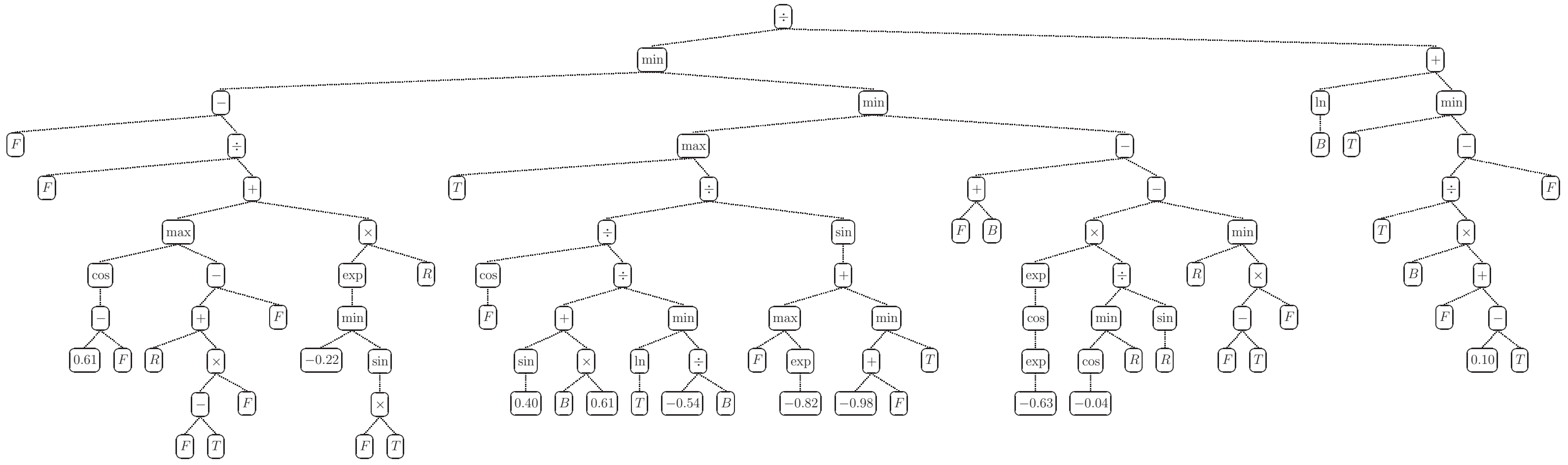


Figure 7.6: The optimal policy with respect to the casualty toll of blue agents.

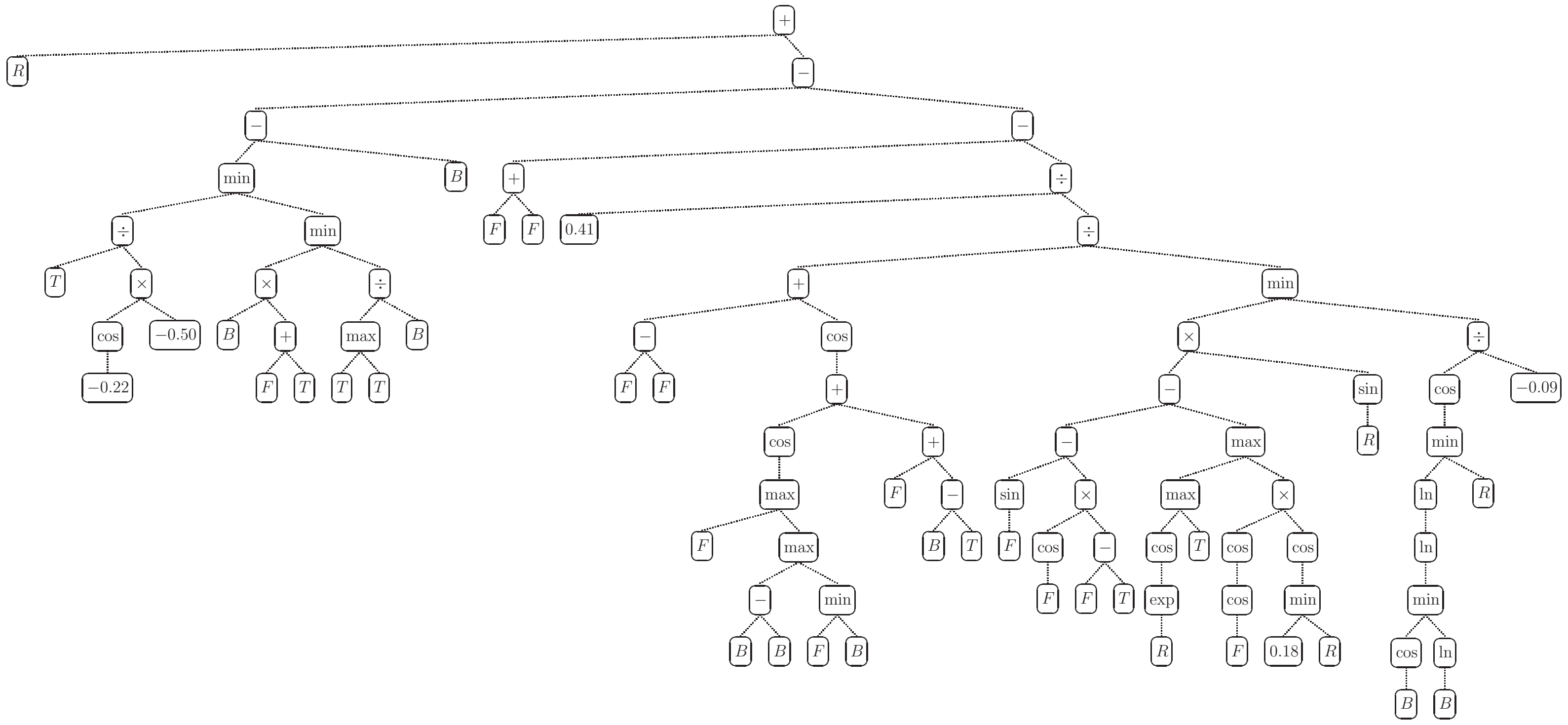


Figure 7.7: The optimal policy with respect to the mission completion time.

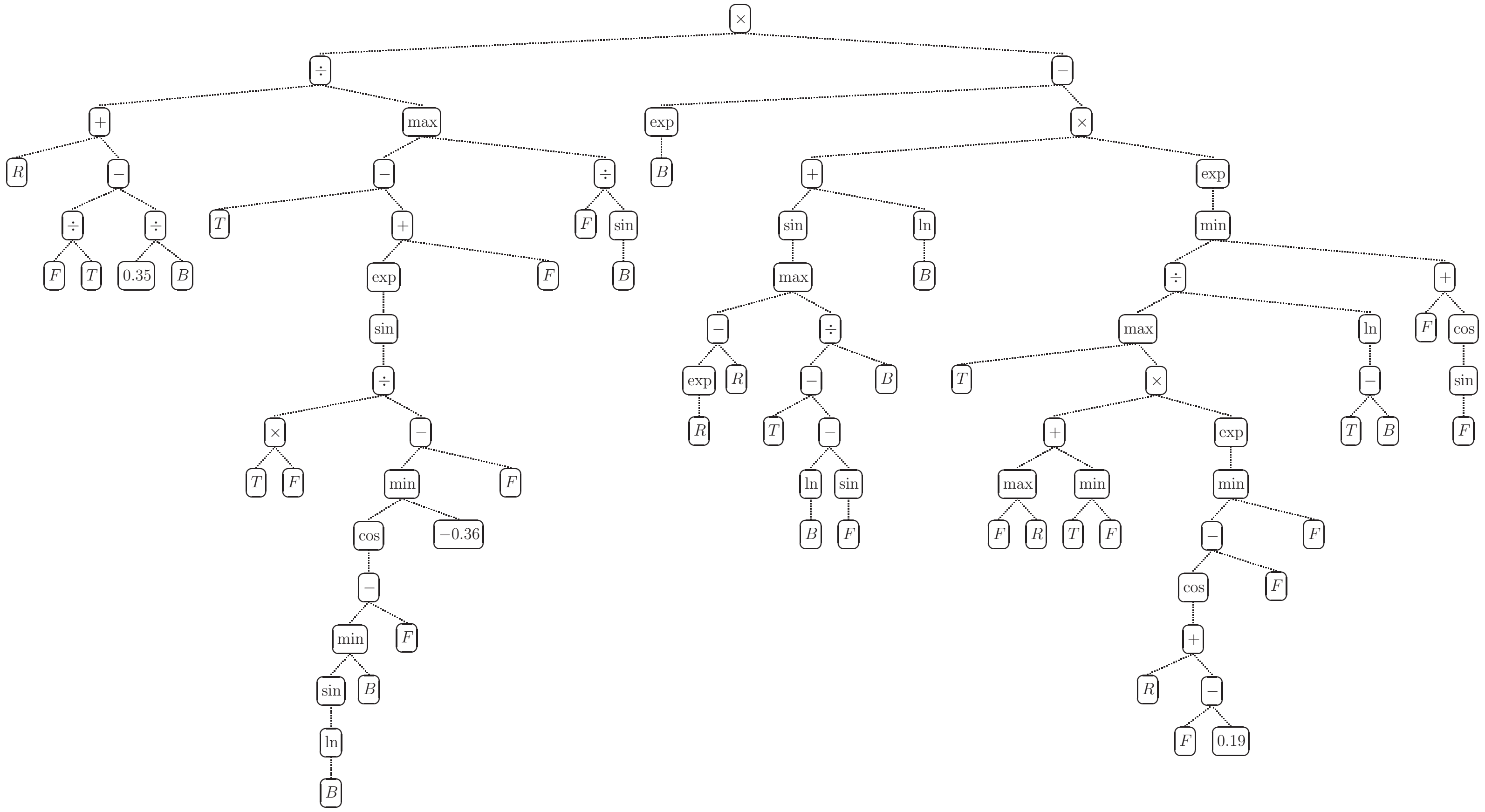


Figure 7.8: The optimal policy with respect to the casualty toll of blue agents given the constraint that the mean of mission completion time has to be within 70 time steps.

Unlike experiments in previous chapters, we demonstrate how simulation can be used to obtain the fitnesses of the policy candidates (feedback on how well they perform), which are then used in turn to steer the search direction. Each policy candidate is plugged into a set of simulated missions. The missions are executed and the outcomes of these missions are measured and used as the fitness of the policy.

The work is a significant deviation from the practice of fitting a policy a priori without the details of the specific mission being taken into account. The concept of "mission-specific policy" seems novel. Of course, our approach assumes that some element of feedback/fitness can be determined via modelling or simulation. This limits the applicability of the technique. However, simulation is used for a great many things in military contexts, e.g., war-gaming. If simulation is good enough for war-gaming strategies, why not for security policies?

By using MOGP (and MOEAs in general), our approach is very flexible, allowing tradeoffs between a variety of criteria to be explored. The criteria chosen are exemplary only; one could imagine tradeoffs being explored between a list of relevant measurements of interest. The Pareto front discovered can reveal useful information about the relationship among different objectives, which may be difficult to obtain otherwise. Such information provides useful insight for policy makers to select and apply the optimal policy that fits the needs of current operational environment on a case-by-case basis. Additionally, it has the merit of deferring the weight assignment that defines the relative importance of each objective after the set of Pareto optimal solutions are discovered. This often makes the weight assignment task easier as the set of Pareto optimal solutions can unfold the relationship between the objectives. One possible avenue of research is to investigate the possibility to extend the heuristic search concept to other parameters used in the simulation. This includes the movement strategy, number of agents, risk budget allocation, etc.

Having said that, the use of Pareto dominance relationship in the fitness evaluation essentially places a practical constraint on the number of objectives that they can cope with. Imagine that there are ten objectives required for optimisation. It is likely that many individuals in the evolving population are non-dominant to one another. The search therefore becomes more or less a random search. This is currently an active research topic in the evolutionary computation community. Indeed, solving an optimisation problem with a large number of objectives is an acknowledged problem generally.

7.6 Conclusions

This chapter first argues that traditional ways of developing security policies are difficult and often inadequate in creating a policy that is optimal in terms of some given objectives. It then attempts to shift the emphasis away from specifying and refining a policy towards searching for a policy that has beneficial and acceptable outcomes from a family of policies. This idea is entirely novel.

We have used a risk-budget based policy family as an example here; it is a means to an end and stands as a proxy for any policy family from which we seek an instance that is best suited to the need of a specific mission. We have demonstrated how GP/MOGP and DOO can be used to search for the Pareto optimal policies. The results show the approaches are very promising.

Chapter 8

Evaluation and conclusions

The work reported in previous chapters provides evidence to support the thesis hypothesis stated in Section 1.3, namely:

> Evolutionary algorithms (EAs) have the potential to be an effective means of determining the security policies that suit dynamic challenging environments.

This chapter reviews the work that has been done, evaluates the extent to which they justify the thesis hypothesis and concludes the thesis by addressing the directions for future work.

8.1 Evaluation

In the previous three chapters, we have detailed several experimentations that serve to support the thesis hypothesis from three different strands of research. We explored the potential of EAs in inferring optimal security policies, dynamically updating security policies with new decision examples and searching for policies with optimal tradeoffs between objectives using simulation runs. This section summarises the work completed in each strand of research and outlines the contributions and novelty of the work presented in this thesis.

8.1.1 Static policy inference

Current security policy is often developed in a top-down approach. High-level security goals are first determined, after which they undergo a series of refinement processes to obtain the low-level executable rules. Although some work has been done in applying machine learning techniques to aid the policy refinement process, there is no previous work to my knowledge in the application of EAs or machine learning techniques in inferring security policies.

Chapter 5 details the experiments in using EAs to infer security policies from decision examples. Here EAs are used as a tool to generalise from a set of low-level examples to a set of high-level rules. Various simple security policies have been attempted and inferred successfully. These include the traditional MLS Bell-LaPadula policy model, the budgetised MLS policy model and the Fuzzy MLS policy model. Two different EAs, namely GP and GE are used. In all cases, the results show that a minimal amount of design effort and domain knowledge are required to infer the reference policy or a close approximation of it. The only requirements are to have a good fitness function and training examples that form a good representation of the target policy.

The last part of the chapter presents how other machine techniques can be incorporated into the policy inference framework created. The fuzzy set concept is used as an example here. Multiple policies are learnt independently. Each focusses on inferring a fuzzy rule for a particular class of decisions (fuzzification). The ultimate output policy, which is an ensemble of all these policies, is formed using a weighted voting mechanism (defuzzification). Various experiments have been carried out to examine different fuzzification and defuzzification techniques. The results show that these approaches can consistently infer policies that closely match with the original reference models used.

An important feature of the approaches investigated is that they can readily handle “noise”, i.e., they can easily accommodate seemingly inconsistent decision making in the training examples. This property would be essential if the technique is to be applied in “dynamic challenging environments” referred to in the thesis hypothesis.

8.1.2 Dynamic policy inference

There will inevitably be times when unseen circumstances demand a decision during operation. In some cases the default automated response may be imperative; in other cases this may be ill-advised. Manual decisions made to override the default one essentially define a new policy. Furthermore, even if the optimal security policy can be developed or inferred automatically, it would eventually become suboptimal due to the changes in either the operational environment or security requirements, or both. Therefore, a security policy has to be able to continually change and be updated to suit the operational needs to maintain its optimality.

Chapter 6 details the experiments on dynamic security policy inference. As there is no dynamic security policy model available and therefore no decision example is available for us to work with, we designed a dynamic security policy model. This model is used to generate time varying decision examples for training and evaluation purposes.

To infer this dynamic security policy model, two novel dynamic learning frameworks based upon MOEAs are designed: one based on Fan's intuition [130] and DOO. In DOO, an n-objective optimisation problem is treated as a $2n$-objective optimisation problem by adding an opposing objective for each of the original objectives. With such a setting, DOO is able to maintain the diversity among the individuals in the population whilst optimising the intended objectives. This diversity can aid in preventing the population from premature convergence and allows the concept drift in the policy to be continually relearnt. The results show that these frameworks are very promising. Reasonably good approximators to the model can be inferred from the examples using these frameworks.

Addressing the need for run-time adaptivity is, we believe, entirely novel. Some degree of adaptability will likely be essential for the dynamic challenging environment referred to in the thesis hypothesis. Our experiments are simple and serve as proof of concept. We are aware that adaptive policies (and automated adaptation in particular) will likely prove to be a controversial area in years to come.

8.1.3 Mission-specific policy discovery

Chapter 7 introduces the notion of mission-specific policy discovery. EAs are used to search for the security policies that can provide the optimal, or at least excellent, tradeoffs among security objectives for a specific mission. Here, EAs serve as an optimisation tool to synthesise the optimal policies, in terms of achieving the mission as well as security objectives without violating the constraints given.

We demonstrate here how simulation can be used to obtain the fitnesses of the policy candidates that are used to guide the policy search. To evaluate the fitness of an individual (policy) for a mission, the policy is first plugged into a simulated mission, then the simulated mission is executed and the outcome of it is measured. This is very different from the practice of fitting a policy a priori without the details of the specific mission being taken into account. This concept of "mission-specific policy" is entirely novel.

Various EA based techniques are used here to discover the optimal policies. These include GP/MOGP and DOO. In all cases, the results show that these techniques are able to discover the set of policies that are optimal for the mission of concern. By using MOGP (and MOEAs in general), tradeoffs between a variety of criteria can be explored. Such information can be valuable to policy makers to select and apply the optimal policy that best fits the current operation. We are unaware of any other work of this nature.

8.1.4 Thesis contributions

In summary, we demonstrate how:

- EAs can be used to infer static security policies from a set of decision examples. Three different ways of representing security policies and two different EAs are investigated. The results show that this idea is feasible.
- the fuzzy set concept can be integrated into the policy inference framework to improve the policy inference performance. The idea is sufficiently generic to be applied to other classification problems, provided that there is a partial ordering among the classes.

- multi-objective evolutionary algorithms (MOEAs) can be used to infer dynamic security policies from a set of decision examples. Two novel dynamic learning frameworks based upon MOEAs are developed: one that is based on Fan's intuition and DOO. Both of them can be used as general dynamic classification algorithms.

- an ensemble policy model can be constructed from multiple models in a single EA run to achieve better performance. The improvement is especially significant in the DOO setting.

- MOEAs can be used to infer a set of Pareto optimal policies that fit a specific mission (or at least a specific family of missions).

- simulation runs can be used in place of a set of decision examples to provide feedback in evaluating the fitness of a policy with respect to the specified high-level objectives.

- MOEAs can be used as a decision making tool where tradeoffs between objectives exist. The Pareto front of the security policies discovered using MOEAs can reveal useful information about the relationship among different objectives, which may be difficult to obtain otherwise. Such information provides useful insight for policy makers to select and apply the optimal policy that fits the needs of current operational environment on a case-by-case basis.

8.2 Envisaged future work

Having discussed the contributions of the thesis, we now outline numerous possible directions for future work that have been identified during the course of this research.

8.2.1 Policy fusion

In dynamic coalitions, parties with different policies can come together to collaborate. Prior to the formation of dynamic coalitions, each party may have its own security policy. An interesting step forward would be to investigate how well EAs could be used to combine these security policies together. One possible way is to generate decision examples from both existing policies and use these examples as the training input for the policy inference framework. MOEAs can also be used to discover the Pareto optimal set of policy candidates, which are then chosen depending on the security requirements. However there are still issues that require further investigation. These include:

- Understanding how to deal with policies that consist of different sets of decision-making factors, which may be measured using different scales.
- Understanding what the implicit priorities that EAs have assigned to the conflicting rules are, what the factors that influence the priorities are and how to control these priorities, etc.

8.2.2 The robustness of a security policy

The framework proposed in this thesis has been shown to be effective in dynamically inferring the optimal policy. However, the optimality of a policy is not always the only factor of concern; the robustness in performance of a security policy in different environments may be equally important. This is especially so in a pervasive operating environment where the deployment of a new policy can be a difficult or expensive process. To incorporate this factor into the proposed framework, a way to quantify the robustness in performance of a security policy is required.

This measure also provides a way to determine the invariant part of the optimal policies for different operational environments of concern. The determination of this invariant part is doubly useful: firstly, it can serve as a template or testing target in the policy development process; secondly, it can help to protect the security policy inference framework from a poisoning attack, which attempts to mislead the inference process in the favour of the attacker by the injection of specially crafted decision examples. (In general, we have assumed that the decision examples used in training have authenticity and the provision of these examples can be accomplished and is outside the scope of this thesis.)

8.2.3 Scalability with the training set size

Scalability is a subtle issue. We have addressed some aspects of this issue. For example, we have shown the method scales well with the size of the training set. In the experiments presented in Chapter 5, we have increased the size of the training set from 100 examples to 1000 examples and the results still remain consistent. Obviously, the fitness evaluation time would increase; 1000 examples take ten times longer than 100 examples to evaluate. This is unlikely to be an issue in practice as the fitness evaluation of each individual can be executed in parallel if necessary. In Chapter 6, we have shown that DOO is able to evolve and update policies with decision examples in an incremental manner. However, there are still some issues remaining with these frameworks that need to be investigated. This includes searching for appropriate techniques to sample old decision examples and examining the generality of the DOO framework.

8.2.4 More complex security policies

The security policies used in this thesis are rather simple. However, note that these policies are either real-world policies or proposals from major research institutes for real world use. They are simple, but by no means "toy" policies. Ultimately, we should strive for simple policies wherever is possible, but at the same time, we should also need to acknowledge that MANET policies may need legitimately to be much more complicated. To cope with complexity, instead of attempting to extract and discover the policies as a whole, we could simply target the areas that need help. Humans produce security policies sequentially too, i.e., they consider in turn authentication policy, file access control, audit policy, etc. In practice, it is also often the case that there are some rules of thumb and constraints that are dictated from on high. We do not need to extract these bits of a policy. Yet, there is still much to answer here, for example:

- Can EAs be used to evolve more complex policies or policies of other types, e.g., obligation policies? If not, how can we divide the security policies into smaller addressable components in a systematic manner?

- How to incorporate the constraints imposed from on high into the policy inference framework to form a continuous learning loop in an efficient manner? Should we take such constraints into consideration in the evolution process? If so, how?

8.2.5 More complex scenarios

The scenario used in Chapter 7 is relatively simple. It has only one type of agent in each team and one type of information. A real test of this approach would be to embed it within a more realistic simulated scenario, with more sophisticated information types, and realistic consequence models. Note that the simulated scenario may be much more complex but we are really interested in some of the measurable properties, which may only be few. For example, how many properties would an operational commander be interested in trading off? The techniques proposed here should be able to scale well with it.

8.2.6 Other deployment domains

Our focus has been on the development of policies for challenging dynamic environments such as military MANETs. However, it may well be that other domains prove also to be suitable deployment domains. For examples, can social network policies be amenable to the techniques described in this thesis?

8.3 Closing remarks

The work reported in this thesis demonstrates a considerable degree of originality supported by extensive experimentation. The case studies are necessarily limited. (MANETs of the sorts we envisage do not really exist right now.) However, the results (published at both optimisation and security venues) demonstrate that inference based approaches using evolutionary algorithms have very considerable promise. Everyone accepts that policy specification is currently difficult, and things are set to worsen as systems are deployed in ever more complex environments with increasing sophistication and subtlety of decision-making process. We recommend these approaches to the research community for further investigation.

References

[1] Common Criteria Project Sponsoring Organisations. *Common Criteria for Information Technology Security Evaluation - Part 1 Introduction and General Model*, August 2005. Version 2.3, adopted by ISO/IEC as ISO/IEC International Standard (IS) 15408 1-3. viii, 16, 19, 20

[2] Pau-Chen Cheng, Pankaj Rohatgi, Claudia Keser, Paul A. Karger, Grant M. Wagner, and Angela Schuett Reninger. Fuzzy Multi-Level Security: An Experiment on Quantified Risk-Adaptive Access Control. Technical report, IBM Research Report RC24190, 2007. viii, 3, 17, 34, 35, 36, 38, 40, 54, 94, 96

[3] Kirk Schloegel, Tom Markham, Walt Heimerdinger, Alberto Schaeffer-Filho, Morris Sloman, Emil Lupu Seraphin B. Calo, and Jorge Lobo. Security Policy Automation — from Specification to Device Configuration. In *Proceedings of the of 26th Army Science Conference (ASC)*, 2008. viii, 50, 51

[4] The MathWorks, Inc., Natick, MA. *Fuzzy Logic Toolbox User's Guide 2*, version 2.2.9 (release 2009a) edition, March 2009. viii, 79, 81

[5] David E. Bell and Leonard J. LaPadula. Secure Computer Systems: Mathematical Foundations. Technical Report ESD-TR-73-278, The MITRE Corporation, Bedford, MA., November 1973. 1, 13, 27

[6] Jonathan D. Moffett and Morris Sloman. The representation of policies as system objects. In *COOCS*, pages 171–184. ACM, 1991. 2, 49, 50

[7] J D Moffett and M S Sloman. User and mechanism views of distributed systems management. *Distributed Systems Engineering*, 1(1):37–47, September 1993. 2, 49, 50

[8] Horizontal Integration: Broader Access Models for Realizing Information Dominance. Technical Report JSR-04-132, The MITRE Corporation JASON Program Office, McLean, Virginia, Dec 2004. 3, 4, 34, 40, 41, 42, 47, 48, 54, 82, 94, 116, 150

[9] Joseph Bonneau, Jonathan Anderson, and Luke Church. Privacy Suites: Shared Privacy for Social Networks. In *SOUPS '09: Proceedings of the 5th Symposium on Usable Privacy and Security*, pages 1–1, New York, NY, USA, 2009. ACM. 5

[10] Annual humie Awards for Human-competitive Results produced by Genetic and Evolutionary Computation. Websites, December 2010. 6, 7

[11] A. Secker, A Freitas, and J. Timmis. AISEC: An artificial immune system for E-mail classification. In R. Sarker, R. Reynolds, H. Abbass, T. Kay-Chen, R. McKay, D Essam, and T. Gedeon, editors, *Proceedings of the Congress on Evolutionary Computation*, pages 131–139, Canberra. Australia, December 2003. IEEE. 6

[12] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security in Computing*. Prentice Hall Professional Technical Reference, 2002. 11

[13] Rick Lehtinen, Deborah Russell, and G. T. Gangemi. *Computer Security Basics*. O'Reilly Media, Inc., 2006. 11, 12, 16

[14] Donn B. Parker. *Fighting Computer Crime: A New Framework for Protecting Information*. John Wiley & Sons, Inc., New York, NY, USA, 1998. 12

[15] D. Gollmann. *Computer Security*. John Wiley & Sons Ltd, 2005. 12, 13, 15, 16, 17, 18

[16] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[17] M. Bishop. *Computer Security: Art and Science.* Addison Wesley, New York, 2002. 12, 14

[18] Andreas Pfitzmann and Marit Köhntopp. Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology. In *International Workshop on Designing Privacy Enhancing Technologies*, pages 1–9, New York, NY, USA, 2001. Springer-Verlag New York, Inc. 12

[19] Catherine Soanes and Angus Stevenson, editors. *The Oxford Dictionary of English.* Oxford University Press, second edition, 2005. 13, 20

[20] K. J. Biba. Integrity Considerations for Secure Computer Systems. Technical Report TR-3153, The MITRE Corporation, Bedford, MA., April 1977. 13, 29

[21] D. D. Clark and D. R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy (SSP '87)*, pages 184–195, Los Angeles, Ca., USA, April 1990. IEEE Computer Society Press. 13, 26, 31

[22] U.S. Department of Defense. Trusted Computer Systems Evaluation Criteria. (Orange Book) 5200.28-STD, National Computer Security Center, Fort Meade, MD, December 1985. 13, 27, 120

[23] Richard E. Barlow and Frank Proschan. *Statistical Theory of Reliability and Life Testing (Probability Models).* International Series in Decision Processes, and Series in Quantitative Methods for Decision Making. Holt, Rinehart and Winston, Inc., 1975. 13

[24] Alan Burns and Andrew J. Wellings. *Real-Time Systems and Programming Languages: ADA 95, Real-Time Java, and Real-Time POSIX.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001. 13

[25] Aleksandar Kuzmanovic and Edward W. Knightly. Low-rate TCP-targeted Denial of Service Attacks: The Shrew vs. The Mice and Elephants. In *Proceedings of the 2003 Conference on Applications, tTchnologies, Architec-*

tures, and Protocols for Computer Communications, SIGCOMM '03, pages 75–86, New York, NY, USA, 2003. ACM. 14

[26] Howard Roberts Chivers. *Security Design Analysis.* PhD thesis, University of York, January 2006. 16, 18

[27] Richard Bejtlich. TaoSecurity — Security is Not Refrigeration. Blog, October 2006. 17

[28] Campbell R. Harvey. Campbell R. Harvey's Hypertextual Finance Glossary. Website, November 2006. 17

[29] Pau Chen Cheng, Pankaj Rohatgi, Claudia Keser, Paul A. Karger, Grant M. Wagner, and Angela Schuett Reninger. Fuzzy Multi-Level Security: An Experiment on Quantified Risk-Adaptive Access Control. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 222–230, Los Alamitos, CA, USA, 2007. IEEE Computer Society. 17, 34, 38, 54

[30] C & A Security Risk Analysis Group. Introduction to Security Risk Analysis. Website, 2005. 17, 18

[31] Red Hat Linux 9: Red Hat Linux Security Guide, 2002. 18

[32] Bruce Schneier. *Secrets & Lies — Digital Security in a Networked World.* John Wiley and Sons, 2000. 19

[33] H. Khurana, V. Gligor, and J. Linn. Reasoning About Joint Administration of Access Policies for Coalition Resources, 2002. 20

[34] Jr. Charles E. Phillips, T.C. Ting, and Steven A. Demurjian. Information Sharing and Security in Dynamic Coalitions. In *SACMAT '02: Proceedings of the seventh ACM Symposium on Access Control Models and Technologies*, pages 87–96, New York, NY, USA, 2002. ACM Press. 20

[35] S.C. Spring, Dennis M. Gormley, K. Scott McMahon, Kenneth Smith, and Daniel Hobbes. Information Sharing for Dynamic Coalitions. Technical Report 2836, Virginia Pacific-Sierra Research, December 2000. 21

[36] E. Royer and C. Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks, 1999. 22

[37] Imrich Chlamtac, Marco Conti, and Jennifer J.-N. Liu. Mobile Ad Hoc Networking: Imperatives and Challenges. *Ad Hoc Networks*, 1(1):13–64, 2003. 22, 23

[38] John Jubin and Janet D. Tornow. The DARPA Packet Radio Network Protocol. 75(1):21–32, January 1987. 22

[39] D.A. Beyer. Accomplishments of the DARPA SURAN Program. In *Military Communications Conference (MILCOM)*, volume 2, pages 855–862, Monterey, CA, USA, September 1990. IEEE. 22

[40] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002. 23

[41] S. Corson and J. Macker. Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. Internet Request for Comment RFC 2501, Internet Engineering Task Force, January 1999. 23

[42] R. Ramanathan and J Redi. A Brief Overview of Ad Hoc Networks: Challenges and Directions. *Communications Magazine, IEEE*, 40:20–22, May 2002. 23

[43] R. Shirey. Internet Security Glossary, Version 2. Technical Report 2828, Internet Engineering Task Force, August 2007. 26

[44] Pierangela Samarati and Sabrina De Capitani di Vimercati. Access Control: Policies, Models, and Mechanisms. In Riccardo Focardi and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*, pages 137–196. Springer, 2001. 26

[45] John McLean. A comment on the "basic security theorem" of Bell and LaPadula. *Information Processing Letters*, 20(2):67–70, February 1985. 28

[46] John McLean. The Specification and Modeling of Computer Security. *Computer*, 23(1):9–16, January 1990. 28

[47] D. F. C. Brewer and M. J. Nash. The chinese wall security policy. In *Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy (SSP '89)*, pages 206–214, Washington - Brussels - Tokyo, May 1989. IEEE. 29

[48] Butler W. Lampson. Protection. In *Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems*, pages 437–443, Princeton University, 1971. 31

[49] David F. Ferraiolo and D. Richard Kuhn. Role-based access control. In *In 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992. 32, 34

[50] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The NIST model for role-based access control: Towards a unified standard. In *Proceedings of the 5th ACM Workshop on Role-Based Access Control (RBAC-00)*, pages 47–64, N.Y., July 26–27 2000. ACM Press. 33, 34

[51] American National Standards Institute, Inc., 25 West 43rd Street, New York, NY 10036, USA. *American National Standard for Information Technology - Role Based Access Control*, February 2004. ANSI/INCITS 359-2004. 33

[52] Jonathan D. Moffett. Control principles and role hierarchies. In *RBAC '98: Proceedings of the third ACM workshop on Role-based access control*, pages 63–69, New York, NY, USA, 1998. ACM. 33

[53] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996. 33

[54] D. Jonscher and K. R. Dittrich. Argos - a configurable access control system for interoperable environments. In *Proceedings of the ninth annual IFIP*

TC11 WG11.3 working conference on Database security IX : status and prospects, pages 43–60, London, UK, UK, 1996. Chapman & Hall, Ltd. 33

[55] M. Nyanchama and S. Osborn. Role-based security: Pros, cons & some research directions. *ACM SIGSAC Review*, 2(2):11–17, June 1993. 34

[56] Roshan Thomas. Team-based access control (TMAC). In *Proceedings of the 2nd ACM Workshop on Role-Based Access Control (RBAC-97)*, pages 13–22, New York, November 6–7 1997. ACM Press. 34

[57] Eve Cohen, Roshan Thomas, William Winsborough, and Deborah Shands. Models for coalition-based access control (CBAC). In *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies (SACMAT-02)*, pages 97–106, New York, June 3–4 2002. ACM Press. 34

[58] Elisa Bertino, Piero A. Bonatti, and Eiena Ferrari. TRBAC: A temporal role-based access control model. In *Proceedings of the 5th ACM Workshop on Role-Based Access Control (RBAC-00)*, pages 21–30, N.Y., July 26–27 2000. ACM Press. 34

[59] A. Tversky and D. Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185:1124–1131, 1974. 47

[60] A. Tversky and D. Kahneman. The framing of decisions and the psychology of choice. *Science*, 211(4481):453–458, 1981. 47

[61] Daniel Kahneman, Paul Slovic, and Amos Tversky, editors. *Judgment under Uncertainty: Heuristics and Biases*. Cambridge University Press, Cambridge, 1982. 47

[62] T. Y. C. Woo and S. S. Lam. Authorization in distributed systems: A formal approach. In *Proceedings of the 1992 IEEE Computer Society Symposium on Security and Privacy (SSP '92)*, pages 33–51, Washington - Brussels - Tokyo, May 1992. IEEE. 48

[63] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26(2):214–260, June 2001. 49

[64] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *Proceedings of the 1997 IEEE Computer Society Symposium on Security and Privacy (SSP '97)*, pages 31–43, Los Alamitos, May 4–7 1997. IEEE Press. 49, 53

[65] Sushil Jajodia, Pierangela Samarati, V. S. Subrahmanian, and Eliza Bertino. A unified framework for enforcing multiple access control policies. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 26(2):474–485, June 1997. 49

[66] Carlos N. Ribeiro, André Zúquete, Paulo Ferreira, and Paulo Guedes. SPL: An access control language for security policies and complex constraints. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS 2001)*, pages 89–107, San Diego, CA, February 2001. Internet Society. 49

[67] Simon Godik and Tim Moses, editors. *eXtensible Access Control Markup Language (XACML) Version 1.0.* February 2003. 49

[68] Morris Sloman and Emil Lupu. Security and management policy specification. *IEEE Network*, 16(2):10–19, March 2002. 49

[69] Morris Sloman. Policy Driven Management For Distributed Systems. *Journal of Network and Systems Management*, 2(4):333–360, December 1994. 49, 52

[70] Jonathan D. Moffett and Morris S. Sloman. Policy hierarchies for distributed systems management. *IEEE Journal on Selected Areas in Communications*, 11:1404–1414, 1993. 49, 52

[71] R. Darimont, E. Delor, P. Massonet, and A. van Lamsweerde. GRAIL/KAOS: An environment for goal-driven requirements engineering. In *Proceedings of the 1997 International Conference on Software Engineering*, pages 612–613. ACM Press, 1997. 50

[72] Emil Lupu and Morris Sloman. Conflicts in policy-based distributed systems management. *IEEE Trans. Software Eng*, 25(6):852–869, 1999. 52, 53

[73] Allan Heydon, Mark W. Maimone, J. D. Tygar, Jeannette M. Wing, and Amy Moormann Zaremski. Miró: Visual specification of security. *IEEE Transactions on Software Engineering*, 16(10):1185–1197, October 1990. 53

[74] John R. Koza. Hierarchical Genetic Algorithms Operating on Populations of Computer Programs. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 768–774, 1989. 56

[75] Conor Ryan, J. J. Collins, and Michael O'Neill. Grammatical Evolution: Evolving Programs for an Arbitrary Language. In Wolfgang Banzhaf, Riccardo Poli, Marc Schoenauer, and Terence C. Fogarty, editors, *Proceedings of the First European Workshop on Genetic Programming*, volume 1391 of *LNCS*, pages 83–95, Paris, April 1998. Springer-Verlag. 56, 63

[76] Charles Darwin. *On the Origin of the Species by Means of Natural Selection.* Murray, London, UK, 1859. 56

[77] Herbert Spencer. *Principles of Biology*, volume 1. Willian and Norgate, London and Edinburgh, 1864. 57

[78] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming — An Introduction: On the Automatic Evolution of Computer Programs and its Applications.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998. 57, 63

[79] John H. Holland. *Adpatation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975. 58

[80] James E. Baker. Adaptive Selection Methods for Genetic Algorithms. In John J. Grefenstette, editor, *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, pages 101–111, Pittsburgh, PA, July 1985. Lawrence Erlbaum Associates. 58

[81] A. Brindle. *Genetic Algorithms for Function Optimization.* PhD thesis, University of Alberta, Edmonton, Alberta, Canada, January 1981. Computer Science Department, Technical Report TR81-2. 58, 124

[82] Dirk Thierens and David E. Goldberg. Elitist Recombination: An Integrated Selection Recombination GA. In *International Conference on Evolutionary Computation*, pages 508–512, 1994. 58

[83] Dirk Thierens. Selection Schemes, Elitist Recombination, and Selection Intensity. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, San Francisco, CA, 1997. Morgan Kaufmann. 58

[84] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing.* Springer, 2003. 59

[85] Wolfgang Banzhaf. Genotype-Phenotype-Mapping and Neutral Variation — A Case Study in Genetic Programming. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature — PPSN III*, pages 322–332, Berlin, 1994. Springer. Lecture Notes in Computer Science 866. 59

[86] Julian F. Miller and Peter Thomson. Cartesian Genetic Programming. In *Proceedings of the European Conference on Genetic Programming*, pages 121–132, London, UK, 2000. Springer-Verlag. 60, 61

[87] David J. Montana. Strongly Typed Genetic Programming. *Evolutionary Computation*, 3(2):199–230, 1995. 61

[88] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992. 61, 66, 86

[89] Markus Brameier and Wolfgang Banzhaf. A Comparison of Linear Genetic Programming and Neural Networks in Medical Data Mining. *IEEE Transactions on Evolutionary Computation*, 5(1):17–26, February 2001. 63

[90] Lee Spector and Alan Robinson. Genetic Programming and Autoconstructive Evolution with the Push Programming Language. *Genetic Programming and Evolvable Machines*, 3(1):7–40, March 2002. 63

[91] Michael O'Neill and Anthony Brabazon. Grammatical Swarm. In Kalyanmoy Deb, Riccardo Poli, Wolfgang Banzhaf, Hans-Georg Beyer, Edmund Burke, Paul Darwen, Dipankar Dasgupta, Dario Floreano, James Foster, Mark Harman, Owen Holland, Pier Luca Lanzi, Lee Spector, Andrea Tettamanzi, Dirk Thierens, and Andy Tyrrell, editors, *Genetic and Evolutionary Computation — GECCO-2004, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 163–174, Seattle, WA, USA, June 26–30 2004. Springer-Verlag. 66

[92] Conor Ryan and R. Muhammad Atif Azad. Sensible Initialisation in Chorus. In Conor Ryan, Terence Soule, Maarten Keijzer, Edward Tsang, Riccardo Poli, and Ernesto Costa, editors, *Proceedings of the Sixth European Conference on Genetic Programming (EuroGP-2003)*, volume 2610 of *LNCS*, pages 394–403, Essex, UK, 2003. Springer Verlag. 66, 102

[93] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001. 71, 123

[94] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature — PPSN VI*, pages 849–858, Berlin, 2000. Springer. 71

[95] Eckart Zitzler and Lothar Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 1998. 71

[96] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Non-dominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994. 72

[97] Man Leung Wong and Kwong Sak Leung. *Data Mining Using Grammar Based Genetic Programming and Applications*, volume 3 of *Genetic Programming*. Kluwer Academic Publishers, January 2000. 74

[98] Roberto R. F. Mendes, Fabricio de B. Voznika, Julio C. Nievola, and Alex A. Freitas. Discovering fuzzy classification rules with Genetic Programming and Co-Evolution. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, page 183, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann. 74

[99] Gisele L. Pappa and Alex A. Freitas. Towards a Genetic Programming Algorithm for Automatically Evolving Rule Induction Algorithms. In Johannes Furnkranz, editor, *ECML/PKDD 2004 Proceedings of the Workshop W8 on Advances in Inductive Learning*, pages 93–108, Pisa, Italy, September 20–24 2004. 74

[100] Tony Brabazon, M. O'Neill, C. Ryan, and J. J. Collins. Uncovering technical trading rules using evolutionary automatic programming. In *Proceedings of 2001 AAANZ Conference (Accounting Association of Australia and NZ)*, Auckland, New Zealand, 1-3 July 2001. 74

[101] Michael O'Neill, Anthony Brabazon, Conor Ryan, and J. J. Collins. Evolving market index trading rules using grammatical evolution. In Egbert J. W. Boers, Stefano Cagnoni, Jens Gottlieb, Emma Hart, Pier Luca Lanzi, Günther Raidl, Robert E. Smith, and Harald Tijink, editors, *Applications of Evolutionary Computing. EvoWorkshops2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM. Proceedings*, volume 2037 of *LNCS*, pages 343–352, Como, Italy, April 18–19 2001. Springer-Verlag.

[102] I. Dempsey, M. O'Neill, and A. Brabazon. Adaptive trading with grammatical evolution. In Gary G. Yen, Simon M. Lucas, Gary Fogel, Graham Kendall, Ralf Salomon, Byoung-Tak Zhang, Carlos A. Coello Coello, and Thomas Philip Runarsson, editors, *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 2587–2592, Vancouver, BC, Canada, 16-21 July 2006. IEEE Press. 74

[103] Tony Brabazon and Michael O'Neill. Trading foreign exchange markets using evolutionary automatic programming. In Alwyn M. Barry, editor, *GECCO 2002: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pages 133–136, New York, 8 July 2002. AAAI. 74

[104] Anthony Brabazon and Michael O'Neill. Evolving technical trading rules for spot foreign-exchange markets using grammatical evolution. 2004. 74

[105] Ajith Abraham and Crina Grosan. Evolving Intrusion Detection Systems. In Nadia Nedjah, Ajith Abraham, and Luiza de Macedo Mourelle, editors, *Genetic Systems Programming: Theory and Experiences*, volume 13 of *Studies in Computational Intelligence*, pages 57–80. Springer, Germany, 2006. Forthcoming. 75

[106] Mark Crosbie and Eugene H. Spafford. Applying Genetic Programming to Intrusion Detection. In E. V. Siegel and J. R. Koza, editors, *Working Notes for the AAAI Symposium on Genetic Programming*, pages 1–8, MIT, Cambridge, MA, USA, 10–12 November 1995. AAAI. 75

[107] Dominic Wilson and Devindar Kaur. Using Grammatical Evolution for Evolving Intrusion Detection Rules. In *ISP'06: Proceedings of the 5th WSEAS International Conference on Information Security and Privacy*, pages 183–188, Stevens Point, Wisconsin, USA, 2006. World Scientific and Engineering Academy and Society (WSEAS). 75

[108] Thomas Weise and Kurt Geihs. DGPF — An Adaptable Framework for Distributed Multi-Objective Search Algorithms applied to the Genetic Programming of Sensor Networks. In Bogdan Filipič and Jurij Šilc, editors, *Pro-*

ceedings of the Second International Conference on Bioinspired Optimization Methods and their Application, BIOMA 2006, pages 157–166, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, 9-10 October 2006. Jožef Stefan Institute. 75

[109] Sevil Şen and John A. Clark. Evolving Intrusion Detection Rules on Mobile Ad Hoc Networks. In Tu Bao Ho and Zhi-Hua Zhou, editors, *PRICAI 2008: Trends in Artificial Intelligence, 10th Pacific Rim International Conference on Artificial Intelligence*, volume 5351 of *Lecture Notes in Computer Science*, pages 1053–1058. Springer, 2008. 75

[110] John A. Clark and Jeremy L. Jacob. Searching for a Solution: Engineering Tradeoffs and the Evolution of Provably Secure Protocols. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 82–95, Washington, DC, USA, 2000. IEEE Computer Society. 75

[111] Hao Chen, John Clark, and Jeremy Jacob. Automated Design of Security Protocols. In *Proceedings of the 2003 Congress on Evolutionary Computation*, pages 2181–2188. IEEE Press, 2003. 75

[112] Hao Chen, John Clark, and Jeremy Jacob. Synthesising Efficient and Effective Security Protocols. In *ARSPA '04: Proceedings of the 1st Automated Reasoning for Security Protocol Analysis Workshop*, pages 25–40, 2004. 75

[113] Julio C. Hernandez-Castro, Juan M. Estevez-Tapiador, Arturo Ribagorda-Garnacho, and Benjamin Ramos-Alvarez. Wheedham: An Automatically Designed Block Cipher by means of Genetic Programming. In Gary G. Yen, Lipo Wang, Piero Bonissone, and Simon M. Lucas, editors, *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 499–506, Vancouver, 6-21 July 2006. IEEE Press. 76

[114] Javier Polimón, Julio C. Hernández-Castro, Juan M. Estévez-Tapiador, and Arturo Ribagorda. Automated Design of a Lightweight Block Cipher with Genetic Programming. *Int. J. Know.-Based Intell. Eng. Syst.*, 12(1):3–14, 2008. 76

[115] David J. Wheeler and Roger M. Needham. TEA, a Tiny Encryption Algorithm. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop*, volume 1008 of *Lecture Notes in Computer Science*, pages 363–366, Leuven, Belgium, 14–16 December 1994. Springer-Verlag. 76

[116] Mark Read. Explicable Boolean Functions. Meng final year project, Department of Computer Science, The University of York, UK, May 2007. 76

[117] Paul Massey, John A. Clark, and Susan Stepney. Human-Competitive Evolution of Quantum Computing Artefacts by Genetic Programming. *Evolutionary Computation*, 14(1):21–40, Spring 2006. Best of GECCO 2004 special issue. 76

[118] P. W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In Shafi Goldwasser, editor, *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Los Alamitos, CA, USA, November 1994. IEEE Computer Society Press. 76

[119] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997. 76

[120] Lotfi A. Zadeh. Fuzzy Sets. *Information and Control*, 8(3):338–353, June 1965. 76, 77

[121] Mark Kantrowitz, Erik Horstkotte, and Cliff Joslyn. Answers to Frequently Asked Questions about Fuzzy Logic and Fuzzy Expert Systems, April 1993. 77, 79

[122] David I. Brubaker. Fuzzy Operators. *EDN*, November 1995. 77, 78

[123] R. R. Yager. On the General Class of Fuzzy Connectives. *Fuzzy Sets and Systems*, 4(3):235–242, 1980. 78

[124] D. Dubois and H. Prade. A Review of Fuzzy Set Aggregation Connectives. *Information Sciences*, 36:85–121, 1985. 78

[125] Hans-Jürgen Zimmermann. *Fuzzy Set Theory and its Applications*. Springer, 4th edition, October 2001. 78

[126] Sean Luke. ECJ A Java-based Evolutionary Computation Research System, 1997. 86, 123, 126, 159

[127] Miguel Nicolau. libge's homepage, 2006. 87

[128] Matthew Wall. GAlib: A C++ library of genetic algorithm components. *Mechanical Engineering Department, Massachusetts Institute of Technology*, 1996. 87

[129] Myles Hollander and Douglas A. Wolfe. *Nonparametric statistical inference*. John Wiley & Sons, New York, 1973. 98

[130] Wei Fan. Systematic data selection to mine concept-drifting data streams. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 128–137, New York, NY, USA, 2004. ACM. 116, 118, 148, 175

[131] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *SIGMOD Rec.*, 34(2):18–26, 2005. 117

[132] Charu C. Aggarwal. *Data Streams: Models and Algorithms (Advances in Database Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 117

[133] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, New York, NY, USA, 2000. ACM. 117

[134] Stefan Bleuler, Martin Brack, Lothar Thiele, and Eckart Zitzler. Multiobjective Genetic Programming: Reducing Bloat using SPEA2. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 536–543, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, May 27–30 2001. IEEE Press. 125

[135] Jurgen Branke. *Evolutionary Optimization in Dynamic Environments.* Kluwer Academic Publishers, Norwell, MA, USA, 2001. 136

[136] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*, pages 231–238. MIT Press, 1995. 143