



Application of the Deterministic Dendritic Cell Algorithm for Multiple Data Stream Analysis in Real- World Domain

Report and Manuals

Denis Pilipenko

5090476

BSc Honours Computing Science Project 2012/2013

Supervisor: George M. Coghill

Declaration

I declare that this document and the accompanying code has been composed by myself, and describes my own work, unless otherwise acknowledged in the text. It has not been accepted in any previous application for a degree. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Signed: Denis Pilipenko

Date: 10/05/2013

Abstract

This research project aims to investigate how the Deterministic Dendritic Cell Algorithm (DDCA) could be applied for multiple data stream analysis in a real-world domain, namely detecting potentially dangerous processes from analysing streams of port scanning data. The Deterministic Dendritic Cell Algorithm, created by Greensmith and Aickelin, belongs to the field of Artificial Immune Systems. It uses a set of dendritic cells (prototypes) to classify normal and anomalous system inputs. The main goal of this project was to expand the existing algorithm that is currently capable of evaluating a single dataset to analysing several data streams and finding co-occurrences between the identified malicious processes. Finally, the effectiveness of the resulting approach was assessed through experimentation and different parameter settings.

The report shows the background research, design, implementation, testing and experimentation with the expanded Deterministic Dendritic Cell Algorithm based on datasets from two different domains (Port Scanning data and SEPA River data).

Acknowledgements

I would like to thank my supervisors, Dr. George Coghill and Dr. Wei Pang, for providing help and support throughout the whole project. Their advice, enthusiasm and knowledge of the field have always pointed me in the right direction with the research.

I would also like to thank Dr. Yaji Sripada for providing additional data that I could use in the project and finding time to answer my inquiries.

Contents

Chapter 1: Introduction	9
1.1 Overview	9
1.2 Motivation	9
1.3 Primary Goals	10
1.4 Secondary Goals	10
Chapter 2: Background	11
2.1 Biological Background	11
2.2 Algorithmic Adaptation	12
2.3 Ideas for Multiple Data Stream Analysis Expansion.....	16
Chapter 3: Related Work	19
3.1 Malware Detection on Windows Operating System.....	19
3.2 Bot Detection	20
3.3 Nmap Port Scan Detection	21
Chapter 4: Problem Domain	23
4.1 Approach	23
Chapter 6: Requirements.....	25
6.1 Functional Requirements	25
6.2 Non-Functional Requirements	26
Chapter 7: Methodology and Technologies	27
7.1 Methodology	27
7.2 Technology	28
Chapter 8: System Design and Architecture	30
8.1 Original DDCA Implementation	30
8.1.1 Initialisation	31
8.1.2 Detection	31

8.1.3 Analysis	32
8.2 Expansion for Multiple Data Stream Analysis	32
8.2.1 Interface and User Input.....	33
8.2.2 Initialisation	33
8.2.3 Detection	34
8.2.4 Analysis	35
8.2.5 Dual Co-occurrence Analysis	35
8.2.6 Multiple Co-occurrence Analysis	36
8.2.7 Presenting The Results	37
Chapter 9: Experimentation and Testing	38
9.1 Port Scanning Data	38
9.2 Experiments.....	39
9.2.1 Experiment 1 – Analysis Integrity Testing	39
9.2.2 Experiment 2 – Multiple Data Stream Analysis Testing	40
9.2.3 Experiment 3 – Dual Co-occurrence Analysis.....	41
9.2.4 Experiment 4 – Multiple Co-occurrence Analysis.....	44
9.2.5 Experiment 5 – Number of Cells in the DC Population	44
9.2.6 Experiment 6 – Migration Threshold.....	46
Chapter 10: SEPA River Data	49
10.1 Data Description.....	49
10.2 Problem Domain	50
10.3 The DDCA Adaptation	50
10.3.1 Signals and Antigens	51
10.3.2 Analysis	52
10.3.3 Dual and Multiple Co-occurrence Analyses.....	52
10.4 Experimentation and Testing	52

10.4.1 MCAV calculation.....	52
10.4.2 Dual Co-occurrence Analysis	53
10.4.3 Multiple Co-occurrence Analysis	54
Chapter 11: Summary and Conclusion	56
11.1 Summary	56
11.2 Future Development	58
11.3 Conclusion	59
Bibliography	61
Appendix A: User Manual	63
Appendix B: Maintenance Manual	68

List of Figures

Figure	Chapter	Page
Figure 2-1: The transition states of a single DC ¹	2: Background and Problem Domain	12
Figure 2-2: Transformation from input to output signals	2: Background and Problem Domain	13
Figure 2-3: The DDCA execution from Initialisation to Analysis ²	2: Background and Problem Domain	16
Figure 7-1: Project Timeline	7: Methodology and Technologies	27
Figure 8-1: Original DDCA program flow	8: System Design and Architecture	30
Figure 8-2: modified DDCA program flow (four data streams)	8: System Design and Architecture	33
Figure 9-1: 'Signal' Input Format	9: Experimentation and Testing	39
Figure 9-2: 'Antigen' Input Format	9: Experimentation and Testing	39
Figure 10-1: SEPA 'Antigen' Input Format	10: SEPA River Data	49
Figure 10-2: SEPA 'Signal' Input Format	10: SEPA River Data	50

List of Equations

Equation	Chapter	Page
Equation 1: CSM calculation	2: Background	13
Equation 2: K calculation	2: Background	13
Equation 3: MCAV calculation	2: Background	15
Equation 4: $K\alpha$ calculation	2: Background	16
Equation 5: MAC calculation	3: Related Work	20

¹ F. Gu, J. Greensmith and J. Aickelin, "Theoretical Formulation and Analysis of the Deterministic Dendritic Cell Algorithm," School of Computer Science, University of Nottingham, NG8 1BB, UK, 2013, p. 5

² F. Gu, J. Greensmith and J. Aickelin, "Theoretical Formulation and Analysis of the Deterministic Dendritic Cell Algorithm," School of Computer Science, University of Nottingham, NG8 1BB, UK, 2013., p. 8

List of Tables

Table	Chapter	Page
Table 6-1: Functional Requirements of the System	6: Requirements	25
Table 9-1: Analysis of the 's1_norm.log'	9: Experimentation and Testing	40
Table 9-2: Analysis of the 's2_norm.log'	9: Experimentation and Testing	40
Table 9-3: Multiple Data Stream Analysis Performance Testing	9: Experimentation and Testing	41
Table 9-4: Results of the Dual Co-occurrence Analysis	9: Experimentation and Testing	43
Table 9-5: Results of the Multiple Co-occurrence Analysis	9: Experimentation and Testing	44
Table 9-6: DC Population Size Testing Results	9: Experimentation and Testing	45
Table 9-7: Migration Threshold Testing Results	9: Experimentation and Testing	47
Table 10-1: SEPA Data MCAV Experiment Results	10: SEPA River Data	53
Table 10-2: SEPA Data Dual Co-occurrence Experiment Results	10: SEPA River Data	54

Chapter 1: Introduction

This section provides an overview of the project and explains the basic principles of the initial approach and ideas for expansion. It contains a list of primary and secondary goals as well as motivation for carrying out this research.

1.1 Overview

Immune Algorithms that belong to Artificial Immune Systems were inspired by the mechanisms of the biological immune system [1]. Similarly, as the host organism is able to protect itself from the threats posed by pathogens and toxic substances, the Artificial Immune Systems have proven to be effective in providing means to detect potentially dangerous processes and operations. This type of behaviour can be referred to as the differentiation of self from unknown and harmful elements. The techniques used In AIS are closely related to machine learning and are often applied to control and optimisation problems.

The Deterministic Dendritic Cell Algorithm belongs to the field of Artificial Immune Systems and focuses specifically on the role and function of the dendritic cells [1]. They are a special type of cells that respond to various forms of danger signals in the system and pass this information to the T-cells responsible for the reactive response. Originally, the DDCA algorithm was designed to analyse data input that consisted of a continuous stream of danger and safe signals coming from the system along with the occurrences of different antigens that interacted with the system. In the end, each antigen received its profile based on the combined analysis of the dendritic cells' collected information.

1.2 Motivation

Over the past twenty years the amounts of data that is being produced by various systems and applications (such as transaction logs, financial data, sensor and network traffic information, logs of system processes, etc.) have increased drastically. Furthermore, many security applications nowadays require processing and combined analysis of separate data streams due to an increased number of complex and distributed attacks. The Dendritic Cell Algorithm has already been effectively used in various applications, mainly for security and threat detection. However, the use of timestamps or any forms of co-occurrence analysis

were often omitted. Introducing these concepts would improve the DDCA capability of detecting complex distributed attacks (e.g. botnet). The Deterministic Dendritic Cell Algorithm could potentially be applied to a wide range of domains and an expansion that allows processing multiple data streams together with the co-occurrence analysis would be a helpful supplement to the initial model.

This idea was greatly inspired by the work of Chris Musselle, specifically his paper “Rethinking Concepts of the Dendritic Cell Algorithm for Multiple Data Stream Analysis” [2]. The author provides a comprehensive overview of the DCA, highlighting its strengths and weaknesses as well as expressing his thoughts on the possible expansion and improvement of the algorithm to better adapt it for the use on real-world data.

1.3 Primary Goals

Here is a list of the primary goals that have to be completed to classify the project as successful:

- Understand the principles behind the Deterministic Dendritic Cell Algorithm and design an appropriate expansion to account for multiple data stream processing. This involves examining documentation and previous work related to the given algorithm as well as getting familiar with the source code selected for modification;
- Extend the analysis to find co-occurrences between data streams;
- Test the resulting application on a dataset from a real-world domain;
- Perform experiments on the dataset to assess the effectiveness of the modified algorithm while making necessary changes to the parameter settings for a better optimisation and performance.

1.4 Secondary Goals

Secondary goals could be added to the project depending on the time remaining before deadline and success of completing all the primary goals:

- Obtain a second dataset from another domain to test the modified algorithm, after experimentation on the initial dataset;
- Write a simple GUI for the application.

Chapter 2: Background

In order to expand and adapt the existing Deterministic Dendritic Cell Algorithm for multiple data stream analysis, an initial background research had to be done to fully understand the concept behind its design. This section provides a more detailed description and analysis of the DDCA as well as discussion regarding its possible expansion.

2.1 Biological Background

Traditionally, the role of the immune system is divided into two subsequent tasks: the detection and elimination of dangerous pathogen. The function of the dendritic cells (DCs), that form the basis of the Dendritic Cell Algorithm, refers to the detection task and it is the first line of defence against intruders. Therefore, dendritic cells are often responsible for policing different substances and tissues or act as inductive mediators for a variety of immune responses [1].

Generally, DCs process two kinds of molecular information, specifically 'signal' and 'antigen'. Signals (safe and danger) are the main indicators of the health status of the monitored environment that are being constantly collected by the dendritic cells. During its lifetime, each DC could exist in one of the three states: 'immature', 'semi-mature' and 'fully mature'. Initially, all DCs start in the immature state, during which they are exposed to a continuous stream of signals while ingesting various substances from their environment. Depending on the combination of signals that were collected, the DC can change into a 'fully mature' state that would alarm the adaptive immune system, or a 'semi-mature' state to suppress it otherwise. For example, if a dendritic cell is exposed to a series of signals generated from a damaged environment, which might indicate the presence of a pathogen, it is more likely to turn into a mature DC. Similarly, if the majority of signals come from a steady and healthy environment, with no signs of danger or damage, then the cell could change into a semi-mature state. Otherwise, the cell keeps on collecting signals until a certain lifespan threshold is reached [3]. The transition from one state to another is shown on Figure 2-1.

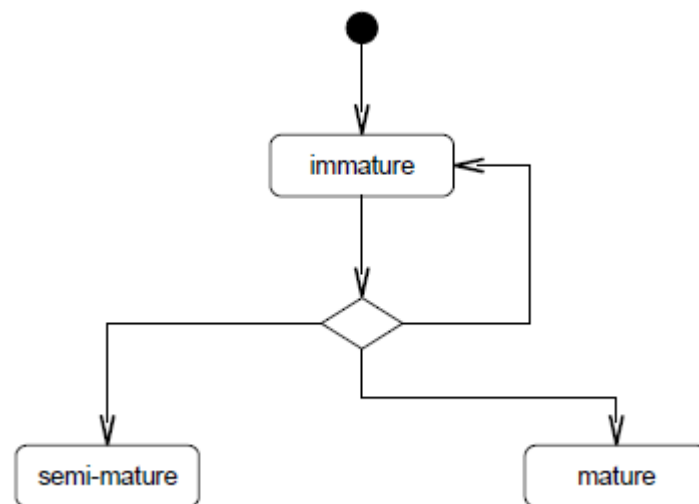


Figure 2-1: The transition states of a single DC

Together with collecting signals, the dendritic cells are responsible for sampling antigens, some of which might have originated from potential pathogens or invading entities. They are subsequently combined with the signals collected from the environment and depending on the DC's state, the adaptive immune system could be instructed to either respond/eliminate or become tolerant towards the given antigen.

2.2 Algorithmic Adaptation

The original DCA was designed and implemented by Julie Greensmith based on the abstract biological model of the dendritic cells [4]. Being a part of the Artificial Immune Systems family, it integrates a concept of the biological dendritic cells' functionality, such as data collection, state differentiation and finding causal correlations. It incorporates two kinds of input data, specifically 'signal' and 'antigen' and assumes that there exists a relationship between these two types. Signals are expressed in real-valued numbers and represent the status of the monitored system, whereas antigens' representation could vary depending on the type of domain used in the application. For example, this could be the ID numbers of system processes, data gathered from different sensors, network traffic information or readings in a robotic system [2].

The current version of the algorithm requires at minimum two signal categories, namely a danger and a safe signal (that occur in the same input). These input signals are being constantly collected by the DCs to produce an output reflecting the overall state of each

individual dendritic cell. This output is later used in antigen analysis and classification. Within the algorithm, the output is represented as 'CSM' (costimulatory output) and 'K' values. Each dendritic cell is allocated a different lifespan limit that defines the amount of input signals the cell is able to process during its lifetime. The CSM signal is designed to reflect the volume of information that is subtracted from the cell's lifespan every time a DC encounters an antigen or receives a signal. At the end of its lifespan, the cell can either change into a 'semi-mature' or a 'fully mature' state, depending on the K signal that indicates the polarisation towards normality or irregularity (danger) [3]. Figure 2-2 displays the transformation of input to output signals.

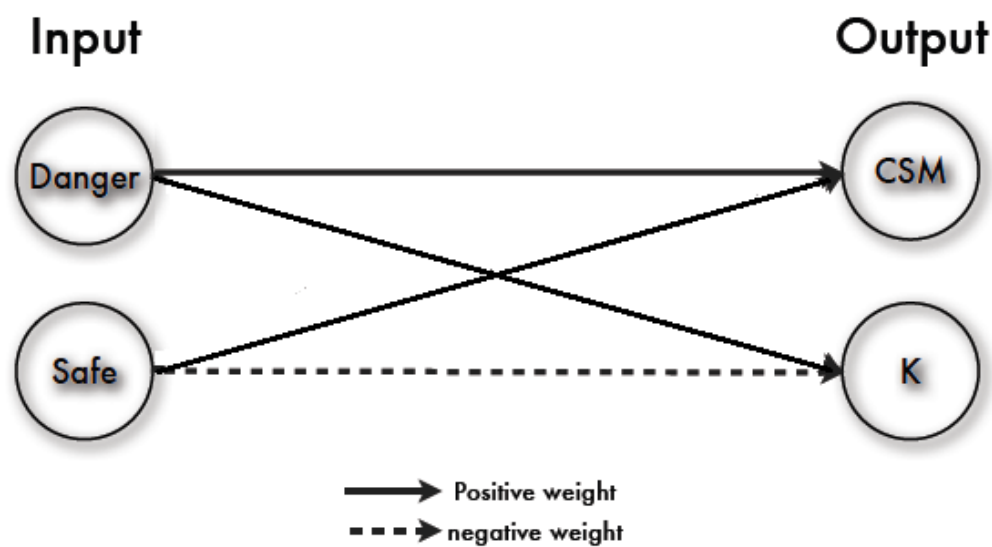


Figure 2-2: Transformation from input to output signals

While both danger and safe signals increase the value of the CSM signal, the K signal is incremented only by danger input and decreased by safe input as shown in Equations 1 and 2 [4].

$$csm = S + D \quad (1)$$

$$k = D - 2S \quad (2)$$

The Deterministic Dendritic Cell Algorithm consists of three main stages:

- Initialisation phase
- Detection phase
- Analysis phase

During the initialisation phase, the DDCA prepares and starts a population of artificial dendritic cells that would later operate as detectors of anomalous processes within the system. Having a different lifespan for every dendritic cell creates a dynamic time-window effect. Furthermore, a uniform distribution of lifespan values allows for the study of this effect in a controllable and repeatable manner [3]. Even though every signal is processed by every DC from the population (with respect to its lifespan), only a specific dendritic cell is selected for the processing of each antigen occurrence. Finally, a global antigen profile is created during this phase (accessible to all the DCs) that is used for the storage and analysis of occurring antigens.

The detection phase begins after the initialisation of the dendritic cells has been completed. All DCs start collecting signals and antigens that occur within the data stream and each cell holds an internal profile for its lifespan, signals and antigens. As it was stated before, with each signal occurrence the dendritic cell's lifespan is subtracted by the CSM value that is calculated by adding together safe and danger values of a signal. The signal profile is defined by the cumulative K value that displays the degree of anomaly (versus normality) encountered by the given cell. For example, having the value of K lower or equal to zero would mean that the cell was mostly present in a safe environment. Conversely, the higher this value gets the higher is the chance that this DC has encountered potentially dangerous elements during its lifetime. The sampled antigens are stored internally within each DC and every time a certain antigen is encountered, its value (number of occurrences) is incremented by one. The signal and antigen collection stops as soon as the dendritic cell's lifespan reaches zero. Afterwards, this DC updates the global antigen profile with respect to the antigens encountered, number of their occurrences and the accumulated K value. Depending on the polarisation of the K value, the encountered antigen's profile could start leaning towards being classified as either safe or dangerous. After all the collected

information has been presented, the DC is returned to its default immature state with most of its internal data being reset back to zero. This way the size of the population is kept constant throughout the whole process [4].

The analysis phase starts after all the data has been processed. To produce the final detection results the algorithm is using information gathered in the global antigen profile. Two new anomaly metrics are introduced for the final output display, namely MCAV and $K\alpha$.

The Mature Context Antigen Value (MCAV) is calculated for every antigen type and, as the name suggests, it measures the proportion of how often the antigen has been classified as dangerous (anomaly) by the ‘fully mature’ DCs as opposed to the total number of times this antigen has been presented by the dendritic cells as shown in Equation 3 [4].

$$MCAV_{\alpha} = \frac{M}{Ag} \quad (3)$$

This anomaly metric can return a value from zero to one and the closer this value gets to one the higher is the chance of that antigen type being anomalous. This normalised metric can be subsequently used for antigen classification by setting an anomaly threshold. However, as it was pointed out in “The Deterministic Dendritic Cell Algorithm” [4] paper, the algorithm does not account for the magnitude of the difference between negative and positive amplitude of the antigen’s K value. For example, during an antigen profiling by a specific dendritic cell the DDCA only checks whether the value of K is greater than zero for the given antigen to be classified as an anomaly and the MCAV metric only accounts for how many times the antigen has been classified as such. This way the K value of -10 would produce the same result as the value of -100 in the calculation of MCAV. This project’s extension of the Deterministic Dendritic Cell Algorithm is mainly focused on using the MCAV value for the co-occurrence analysis and this small deficiency did not seem to affect the overall performance. However, it might be beneficial to incorporate this information in the future to create a more comprehensive and sophisticated metric. It is important to mention that the antigen could be considered dangerous even with a negative K value, as there might have been many cells that had a positive (but small) K value indicating danger that were overwritten by some of the larger negative K values of the semi-mature cells.

K_α is another metric implemented to characterise the resulting antigens. It shows the average K value of each antigen, which is based on the gathered information from all the dendritic cells that have encountered this antigen type. The K_α metric complements the MCAV value by showing how big is the tendency towards normality or anomaly for each specific antigen α . The K_α value is calculated by dividing the accumulated K readings by the total numbers of times the antigen has been encountered by the DCs as shown in Equation 4 [4].

$$K_\alpha = \frac{\sum_m k_m}{\sum_m \alpha_m} \quad (4)$$

Overall, the DDCA runs in a way so that the initialisation and analysis phases are performed at the population level while signal collection and antigen profile updates are executed at a single dendritic cell's level. This process is illustrated in Figure 2-3.

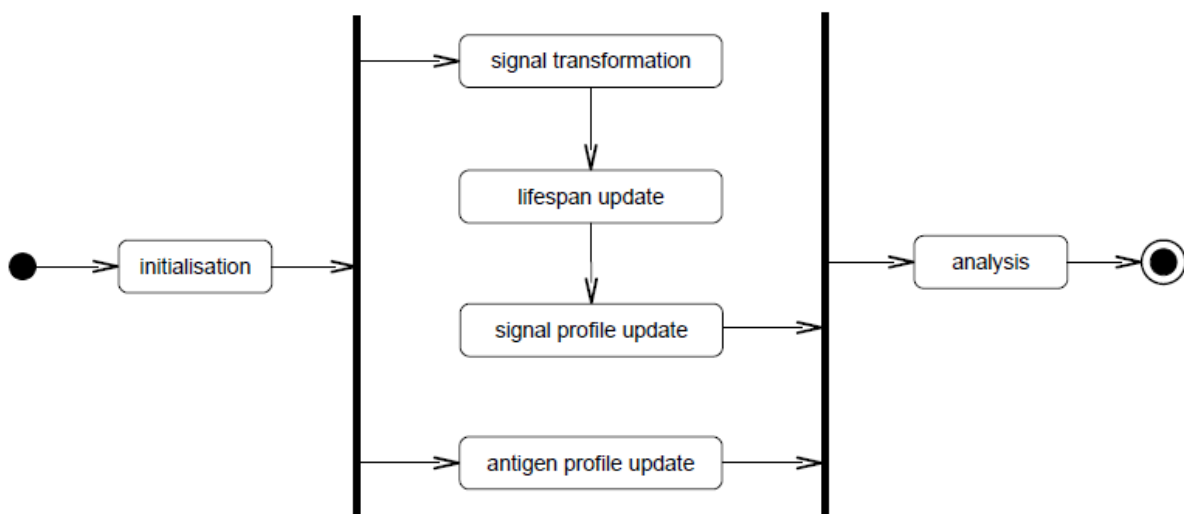


Figure 2-3: The DDCA execution from Initialisation to Analysis

2.3 Ideas for Multiple Data Stream Analysis Expansion

The Deterministic Dendritic Cell Algorithm explained above is capable of analysing a single data stream of signals and antigens at a time. Extending the algorithm to be able to process several data streams would not only improve its efficiency but could allow branching out to new areas and forms of analysis. This could be particularly useful considering large volumes of data that are being produced by many modern systems and applications and the need for cross-stream analysis.

Chris Musselle, in his paper “Rethinking Concepts of the Dendritic Cell Algorithm for Multiple Data Stream Analysis” [2], describes the strengths and weaknesses of the Dendritic Cell Algorithm and discusses a redefined concept of the DCA that would be capable of a multiple data stream analysis. Even though the DCA has previously been applied to many specific problems, the author is raising concern regarding its utility in the real-world domain applications. It is worth mentioning that the Deterministic Dendritic Cell Algorithm is an altered model of the original DCA. The later version of the algorithm preserved all the main principles of its predecessor, although it was made deterministic by removing stochastic components. These changes were made for the ease of analysis, better control over parameterisation (as well as the overall process) and to provide a more formal and quantifiable definition for both signals and antigens.

Musselle based his work on the original DCA. Even though many limitations have been fixed in the following version, he was still unsatisfied with the concept of using ‘signals’ and ‘antigens’ in specific real-world applications. He also argued that much simpler techniques could have been applied to a number of situations where the DCA was previously used, due to its fairly complex design with many interrelating components and parameters. Therefore, the aim of that paper was to expand the concept of DCA by rethinking the ideas of ‘signals’ and ‘antigens’ for a specific application that would support multiple data stream analysis.

To implement this idea, a completely new algorithm was introduced, namely the Change point Detection Subspace Tracker (CD-ST). Its main purpose was to detect substantial change points across multiple data streams. The second step was to redefine the concepts of ‘signals’ and ‘antigens’ and combine them with the new CD-ST algorithm for a novel anomaly detection framework.

The Change point Detection Subspace Tracker (CD-ST) works by compressing data input streams into a reduced representation. It flags any unexpected change points and updates the representation with every new data point through subspace tracking algorithm. The CD-ST uses the Symbolic Aggregate approximation technique (SAX) for the symbolic interpretation of time series data. The SAX representation requires defining two parameters, specifically the word length which is the number of characters the time series data is going to be reduced to and a set of different characters that would represent various

states or changes within time series. For example, a SAX 'word' of length 10 and a character set of {a,b,c,d} could be visualised as {a a b c d d c b d d}.

Furthermore, SAX representation was similarly used for redefining the concepts of signals and antigens. The points of change in this model are viewed as the likely indicators of anomalous behaviour within multiple data streams. Therefore, a SAX snapshot of all data streams around the point of change has been selected as a new representation for the term 'antigen'. On the other hand, the concept of 'signal' can vary from one application to another. As the main task of a signal is to reflect the overall system status or, in other words, to become a heuristic indicator for antigen classification, the idea of this model was to redefine it as an appropriate feature that could be retrieved from the CD-ST algorithm or SAX representation output. This feature is selected based on its ability to identify whether the change point observed should be treated as a threat or normality.

This model was tested on a dataset of internet traffic measurements taken from a data centre. The CD-ST algorithm was proven effective in picking up both single and multiple stream change points. A set of 'antigens' was successfully created by the SAX snapshot representation during these points of change. However, difficulties arose when it came to selecting the new 'signal' definition. Even though the model was quick and efficient at locating change points in data series, further research is needed to investigate which features of the SAX snapshot representation are able to correctly identify whether the change could be considered genuine or anomalous.

To summarise, this paper provided a valuable groundwork and inspiration to expand the Dendritic Cell Algorithm for multiple data stream analysis in real-world applications. Many key decisions were made based on the analysis of Chris Musselle's work, specifically:

- After examining the original DCA and the modified Deterministic Dendritic Cell Algorithm, it was decided not to redefine the concepts of 'antigens' and 'signals' so radically. However, some hand tuning is inevitable when applying the algorithm to a novel real-world domain
- The concept of change points used in this paper could provide a new framework for DDCA expansion even within its original design
- The use of timestamps would be required for implementing these ideas

Chapter 3: Related Work

To gain a greater understanding of the Dendritic Cell Algorithm, its strengths, weaknesses and practical applications, I have taken some time to study previous works and researches where the algorithm has been tested on the real-world data. Although many previous projects were based on a single data stream analysis, many valuable conclusions could be made that might help to develop and expand the DDCA further.

3.1 Malware Detection on Windows Operating System

In the paper “A Sense of ‘Danger’ for Windows Processes” [5] S. Manzoor, M. Zubair Shafiq, S. Momina Tabish and M. Farooq investigate the suitability of the Dendritic Cell Algorithm, both classical and deterministic, for malware detection at run-time. For the experimentation setup, a set of API call traces of real malware and standard Windows processes were collected. The efficiency of DCA in terms of malware detection and classification was then evaluated.

The results obtained showed the potential of the DCA in the domain of run-time malware detection with the Deterministic Dendritic Cell Algorithm showing superior results in terms of classification accuracy compared to the classical version. Furthermore, this study also investigated the effects of antigen multiplier and moving time-windows on the performance of the DCA.

The *Antigen Multiplier* concept is occasionally implemented in the cases when the datasets used for analysis contain only one or a small number of copies of each antigen type. This might lead to a scenario when a certain antigen is only sampled by a few dendritic cells, which might not be sufficient to generate an accurate MCAV value in the end. The *Antigen Multiplier* concept works by copying each antigen type multiple times to allow sampling by a larger DC population. This averages the classification decision and helps to improve the overall accuracy of the algorithm.

The *Moving Time-Windows* concept was also inspired by the mechanisms of the biological immune system. The signals in our body do not disappear suddenly but rather fade away over time. This has suggested a concept in the DCA where the new signals become the

average of the old signals within a particular time-window. This approach tends to reduce the noise in input signals.

This results from this study showed that the antigen multiplier had a positive effect on the malware detection accuracy while moving time-windows did not seem to affect it significantly.

3.2 Bot Detection

The problem of complex distributed attacks has become particularly widespread in recent years. A common example is botnet, a distributed, decentralised network of subverted machines, controlled by an attacker. A host machine can become a part of the botnet by accidentally installing a piece of malicious software, called a bot. The paper “DCA for Bot Detection” [6] evaluates the capability of the Dendritic Cell Algorithm to detect the presence of a single bot on a compromised machine.

The bots are usually communicating through traditional networking protocols such as IRC, HTTP and more recently P2P. Many types of bots also share common behavioural attributes like keylogging and network packet flooding. Therefore, various communication functions, file access functions and keyboard state functions were selected for the ‘antigen’ definition. For the ‘signal’ definition, the time difference between receiving and sending data across the network was selected. Previous observations showed that the bots have responded immediately to the central attacker’s commands whereas standard chat and protocol communications generally have a time delay. Moreover, a novel antigen coefficient, termed MAC, was incorporated in the algorithm. As it had been shown previously, the low number of antigens per antigen type can lead to an inaccurate MCAV value. In this paper, the researchers addressed the problem by introducing the MCAV Antigen Coefficient or MAC. This value is calculated by multiplying the MCAV value of an antigen by the number of this antigen’s occurrences and divided by the total number of occurrences of all antigens as shown in Equation 5 [6].

$$MAC_x = \frac{MCAV_x * Antigen_x}{\sum_{i=1}^n Antigen_i} \quad (5)$$

The experimental results showed that the DCA was capable of distinguishing between normal and bot processes on a compromised machine. Additionally, the introduction of the MAC value has increased the accuracy of the results. An interesting observation was made regarding the weights for the safe signals. It appears that higher values for the safe signal weight tend to reduce the potential false positives without producing false negative errors.

3.3 Nmap Port Scan Detection

The paper “Dendritic Cells Algorithm and Its Application to Nmap Portscan Detection” [7] is another example of the DCA application in a real-world domain. The aim of the work of Fang Xianjin and Song Danjie was not only the evaluation of the general applicability of the DCA for Nmap port scan detection but also the demonstration of the feasibility and robustness of the algorithm, and its responsiveness to the change of different parameters in a series of experiments. This work is an extension to a previous research of port scan detection by Greensmith, Aickelin and Twycross [8].

The Nmap port scan attack might allow a remote intruder to get access to a subnet of machines (running within a similar IP address range) if one of these machines has been subverted remotely via ssh (secure shell). For the DCA setup, the concept of ‘antigen’ has been mapped to data containing attack behaviours, specifically the processes that run within a ssh session. Some of these processes are normal while the others could be dangerous. The ‘signals’ were mapped to the number of network packets sent per second (danger indicator) and their rate of change (safe indicator). To test the sensitivity of the DCA to various parameter changes, four different parameter settings were experimented with, namely the size of the DC population, the size of the antigen vector (number of different types of antigen), the number of the DC antigen receptors (how many cells process a single antigen occurrence) and the overall number of antigens presented.

Experimental results showed that the DCA was effective in detecting anomalous port scan activity. High true positive rates were observed for the detection of dangerous activity as well as low false positive rates for the detection of benign processes. Even though the DCA has many different parameters, the results showed that the algorithm was not particularly sensitive to their changes, given that the values are within a sensible range. This highlights

the robustness of the DCA algorithm and shows that in many cases the satisfactory results could be achieved with its default settings.

Chapter 4: Problem Domain

The Dendritic Cell Algorithm has already been effectively used in various applications and domains, mainly for security and threat detection. This project mainly focuses on expanding the analysis of port scanning data used by Greensmith, Aickelin and Twycross in their paper “Articulation and Clarification of the Dendritic Cell Algorithm” [8]. The algorithm has been successful at identifying anomalous antigens/processes that could essentially be a part of a larger scale distributed or botnet attack. When a single machine on a network is subverted, it might start sending packets and malicious content to other machines within the same network, which could lead to the expansion of the distributed attack. Furthermore, when a network of subverted machines receives a command from the central attacker, this could lead to a series of malicious actions from a number of machines that occur at the same time. There is a high chance of these processes happening within a certain time window and detecting their co-occurrence on different machines could supplement the initial profiling and point to the most common elements of the attack.

In case of success, this approach could be applied to other domains where co-occurrence of various events might imply a potential danger or simply indicate a set of important interrelations.

4.1 Approach

This is a research project aimed at designing and evaluating a possible expansion to the deterministic version of the DCA. The new concept incorporates all the previous features and analytical capabilities of the original model and the newly developed co-occurrence analysis of multiple data streams is based primarily on the initial analysis of each data stream. As it was mentioned before, many ideas and design inspirations were obtained from the work of Chris Musselle. Although Chris Musselle completely rethinks the concept of ‘signals’ and ‘antigens’ by defining them as snapshots of SAX representation, it was decided to continue working with the original implementation. Even though SAX provides a comprehensive representation of data change points that is subsequently used for antigen and signal definition, further research is needed to correctly identify which features of SAX snapshots could be used for signal categorisation. Therefore, in this project the ‘signal’ is still defined as a numerical representation with two categories (safe and danger) of the

system status, whereas ‘antigens’ are represented by the process ID numbers of system calls that occur during a session.

The concept of capturing data change points that occur during sessions provided a good starting framework for expanding the algorithm for multiple data stream analysis. The paper “Rethinking Concepts of the Dendritic Cell Algorithm for Multiple Data Stream Analysis” [2] describes the use of the CD-ST (Change point Detection Subspace Tracker) algorithm to detect substantial change points across multiple data streams. As we are not using SAX representation in our application, this concept was slightly altered to take into account the original ‘signal’ and ‘antigen’ definitions. In our model, timestamps are used to keep track of the antigen occurrences and the situation is considered particularly unsafe when two or more dangerous antigens tend to appear at the same time window, whether this happens within a single or multiple data streams.

As this is a research project for testing the efficiency of the new methodology, the application uses a simulation (offline) rather than a real-time analysis. Firstly, most DCA and DDCA applications were implemented using the offline analysis and post-processing the data did not seem to affect the overall efficiency of the system. Secondly, the datasets for the analysis in this project are contained in a series of log files and choosing the simulation would provide more time for the development of the new forms of analysis and performance testing.

The main analytical addition to the algorithm was the implementation of methods for co-occurrence checking between multiple streams of data (or several data logs in this case). For this reason, two various but interrelated forms of co-occurrence were used, namely Dual and Multiple co-occurrences. Dual Co-occurrence extracts the pairs of dangerous antigens and shows how closely they are related. Multiple Co-occurrence, however, uses the information of every interconnected harmful element to produce a set of several (more than two) antigens that are likely to occur together and pose a more widespread threat.

Chapter 6: Requirements

This section describes the requirements for this project. Table 1 shows the functional requirements, highlighting the desired functionality from a technical side. Afterwards, there is a discussion of non-functional qualities of the system and the anticipated outcome of the modified algorithm. Table 6-1 presents the functional requirements.

6.1 Functional Requirements

No.	Description	Priority
1	Analysis	HIGH
1.1	Preserve the original features of the DDCA analysis	
1.1.1	Initialisation of a DC's population	HIGH
1.1.2	Processing of signal and antigen input	HIGH
1.1.3	DC: Updating the K and CSM values based on signal input	HIGH
1.1.4	DC: Updating the local and global (at the end of lifespan) antigen profiles	HIGH
1.1.5	Calculating the K and MCAV values of each antigen	HIGH
1.1.6	Presenting the results (antigen profiles: ID, K, MCAV)	HIGH
1.2	Expanding the DDCA for multiple data stream analysis	
1.2.1	Processing multiple data streams in one run	HIGH
1.2.2	Performing the initial analysis on every stream	HIGH
1.2.3	Antigen profile: collecting timestamps	HIGH
1.2.4	Perform 'Dual' co-occurrence analysis	HIGH
1.2.5	Perform 'Multiple' co-occurrence analysis	HIGH
1.2.6	Refine the results output (include the new analysis)	HIGH
2	User Input	
2.1	Allow the user to select the number of data streams	HIGH
2.2	Allow the user to enter the names of the data files	HIGH
2.3	Allow the user to select the size of the DC population	HIGH
2.4	Allow the user to select the migration threshold	HIGH
3	Output format	
3.1	Display the results during run-time	HIGH
3.2	Save the results in a separate file	HIGH

Table 6-1: Functional Requirements of the System

6.2 Non-Functional Requirements

As this system is an extension of a previous DDCA implementation by Julie Greensmith (with modification by Feng Gu), it was crucially important to devote a considerable amount of time to testing. This was to ensure that the alterations did not affect the overall performance of the algorithm and the quality of its analysis.

The system's scalability is another important aspect. As the algorithm would now be analysing several streams of data instead of one, the process should require more computational resources as well as be more time consuming. Furthermore, running the co-occurrence analysis requires storing additional values, which could also affect the system's performance. It is important that the algorithm handles the new functionality and data load well.

Moreover, running the DDCA requires setting up various parameters and thresholds for data analysis (size of the DC population, migration threshold, settings for co-occurrence analyses, timestamp collection). These parameters need to be optimised for the algorithm to be computationally effective and produce an accurate analysis.

Chapter 7: Methodology and Technologies

This chapter describes the methodology used in this project for the research, design, implementation and testing. It also tells about the technologies used to achieve these goals.

7.1 Methodology

This section is essentially an extension to the project plan provided during the first week of work. At that time, an approximate guideline to follow throughout the project was set, highlighting the deadlines for various project stages. However, even though the work was generally following the original guideline, some stages took longer than was initially planned as their thorough completion was found to be particularly important for the further progression. Figure 7-1 shows a revised timeline of the main project activities.

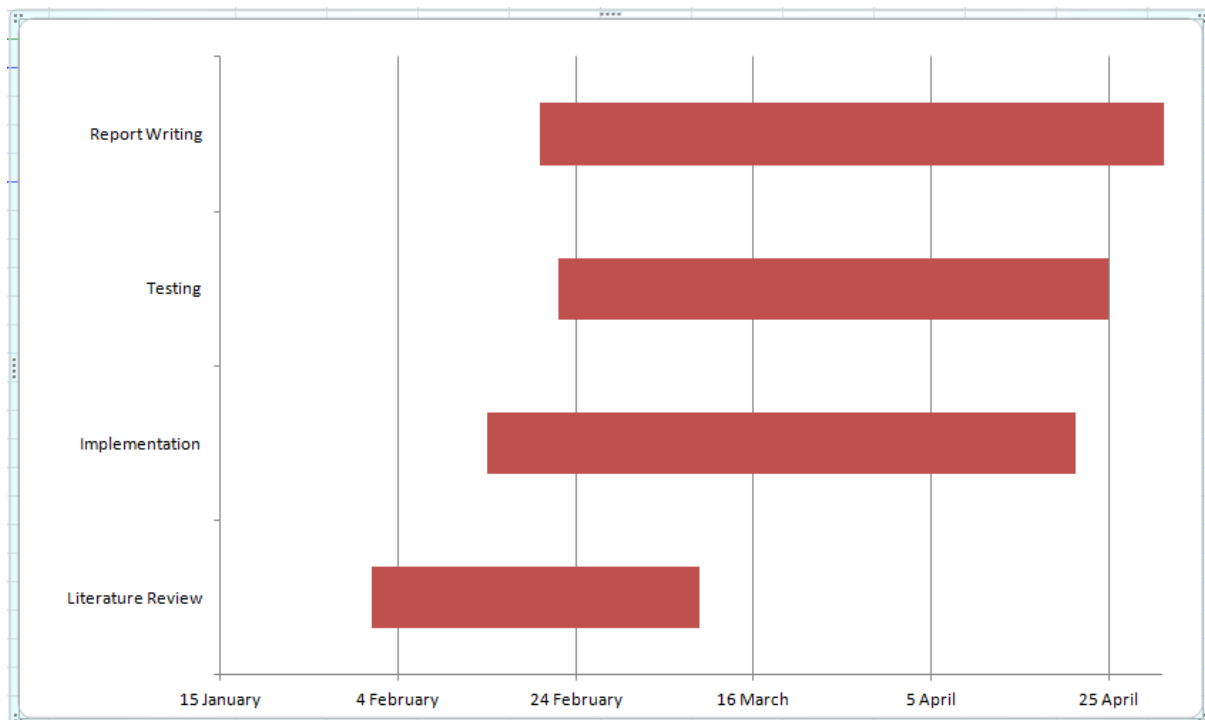


Figure 7-1: Project Timeline

The work on the project started from researching relevant literature and getting familiar with the concept of the Artificial Immune Systems. As this was a completely new field to discover, comprehensively understanding the model of the Deterministic Dendritic Cell Algorithm was critically important. Moreover, sole study of the DDCA concept was not sufficient for the completion of the project and considerable amount of time was spent researching previous applications of the algorithm. Finally, as this system is an extension of

a previous implementation, the remaining time was used to study the code well enough to be able to make modifications to it. Further references to the literature and related work were also consistently made throughout the project.

The implementation stage consisted of designing the future system and the subsequent realisation of these plans. Many key design decisions had to be made during this stage, and the solutions were often obtained from the analysis of previous work. Many new ideas have been originating with the project's progression, which has extended the implementation process until the last weeks of April.

The testing phase has started shortly after the beginning of implementation. A constant testing after every major modification was required to ensure that the changes have not affected any previous functionality. The testing stage also included experimentation with the real data on a complete system. These tests were conducted to evaluate the efficiency of the new co-occurrence analysis, the ability to process several streams in one run and the effects of different parameter settings on the performance of the algorithm and overall analysis accuracy.

The writing of the report was kept rather flexible, with various sections of the paper being written throughout the entire working period.

7.2 Technology

The use of technology was mostly determined by the format in which the original version of the Deterministic Dendritic Cell Algorithm was implemented. For this reason, the C programming language was selected for the implementation of the current system. Another possibility would have been Ruby. However, due to several aspects it was not chosen over the C language. Firstly, the implementation in Ruby represented a more generalised version of the DDCA; the control over parameterisation was not as precise as in C. Secondly, my experience and knowledge of the C language is superior to that of Ruby and using familiar tools is one of the strongest recommendations for a developer. Therefore, it was decided that C would be a better option for this project's development.

The next step was deciding which Operating System to use for this project. As the C language is platform independent, the main choice was between Linux and Windows. The

implementation did not require many specific libraries or additional applications, therefore the two systems were equivalent in terms of compiling and running. Windows OS was chosen in the end as it provided more freedom in terms of workplaces without the necessity of installing any additional software.

Initially, the work was started on a university classroom PC (Intel(R) Core™ i3-2100 CPU @3.10 GHz, 3 Gb RAM). However, after further development the machine has started to run slower occasionally, which often varied depending on the PC used. The development was then continued on my own machine (Intel(R) Core (TM)2 Solo CPU @ 1.40 GHZ, 4 Gb RAM) where the performance remained constant and a full control of the system resources and processes was available.

Code::blocks was chosen as the IDE for this project. Dev-C++ was selected at first but due to an old C compiler and lack of overall maintenance the work was continued in a more robust and dependable environment.

Chapter 8: System Design and Architecture

The following chapter describes the system's basic design and its underlying program flow. As this model is an extension of the original DDCA implementation, the first subsection will outline the core architecture of that implementation in order to provide a better explanation of the changes made in the new algorithm. The second section explains the design and ideas behind the system's modification as well as the new forms of multiple data stream analysis.

8.1 Original DDCA Implementation

Julie Greensmith has implemented this version of the algorithm on 27/03/2008 and it was later modified by Feng Gu on 11/07/2008 to integrate the functions of calculating the MCAV and $K\alpha$. It was specifically designed to work with the port scanning data from the Port Scan Experiment [8]. The algorithm exhibits all the main functionality and properties of the Deterministic Dendritic Cell Algorithm described in the previous chapters. This section provides a more detailed definition of the system's program flow, explaining each individual segment from a technical perspective. The overall program flow is illustrated in Figure 8-1.

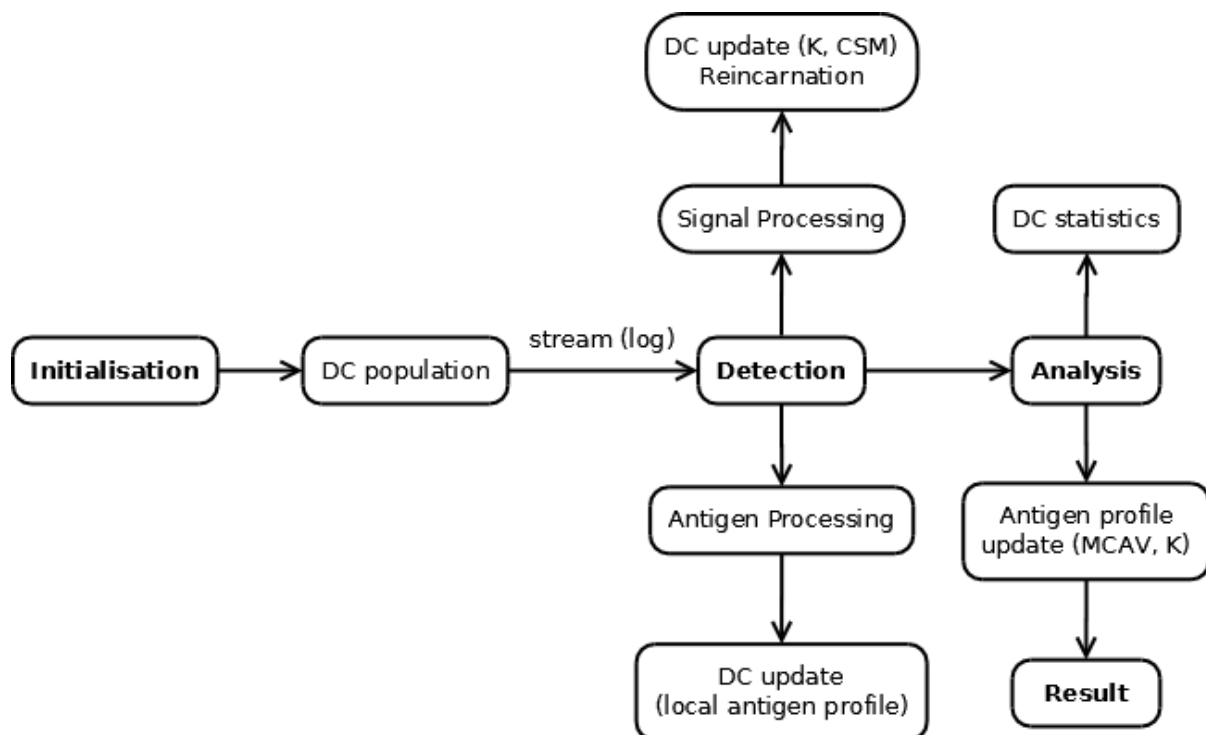


Figure 8-1: Original DDCA program flow

Initialisation, Detection and Analysis are the three main phases of the algorithm. The two central parameters (affecting analysis), namely the size of the DC population (NUM_CELL) and the maximum DC lifespan (MAX_MIG), are predefined in the beginning of the code and set before the Initialisation. A global antigen profile of a fixed size is initialised at the start of the algorithm.

8.1.1 Initialisation

The main purpose of this stage is to create a population of dendritic cells for the analysis of incoming signals and antigens. Each cell is initialised in a similar manner. The cell's ID is represented by an integer that is allocated incrementally. The DC's lifespan is distributed uniformly across the whole population. Statistical variables such as the number of iterations of signal updates received, the number of incarnations, the K value and the total amount of antigen a DC has collected are all set to zero. Furthermore, every cell holds a local antigen profile that stores the number of times each antigen type has been encountered. After the dendritic cells have been initialised, the algorithm moves on to detection phase.

8.1.2 Detection

This model was designed to receive the signals manually through the user input. Each input is processed and can be identified as either a 'signal' or an 'antigen' depending on the second element in the buffer.

In case of an 'antigen' input, one cell is selected for collecting that antigen type. Every new antigen occurrence is processed by a different cell that is also determined incrementally. The selected cell updates its local antigen profile by changing the total number of times this particular antigen type has been encountered.

On the other hand, every 'signal' input is processed identically by all the dendritic cells in the population. The aim of signal processing is to update the state of every DC according to the current system status. Each 'signal' input carries two distinct values (float), namely safe and danger indicators. This data is used for the calculation of CSM (safe signal + danger signal) and K (danger signal – 2 safe signals) values, which is showed in Figures 2 and 3. The resulting values are then passed to the dendritic cells to update their lifespan by subtracting the CSM value from the cell's lifespan and adding the signal's K value to the local K indicator. When a DC's lifespan reaches zero, all the antigen types that this DC has encountered get

their global profile updated according to the cell's accumulated K value (more on this in the Analysis section). After this procedure, the DC is reinitialised ("reincarnated") with the same parameters and values as in the Initialisation phase, except for some of the statistical data such as the total number of iterations and incarnations. The Analysis stage starts once all the data has been processed.

8.1.3 Analysis

At the beginning of the Analysis phase, all DCs in the population undergo the same procedure as when a cell reaches its lifespan limit during the Detection phase, specifically the antigen log that updates the global antigen profile. During the antigen log each cell iterates through its collected antigens list and updates their K value (in the global antigen profile) by adding to it (or subtracting, depending on the sign) its local accumulated K value. This process is repeated the number of times this antigen type has been collected by the given cell. The resulting value could either be positive (representing danger) or negative/zero (indicating normality).

Furthermore, each antigen has a counter of anomaly (m) and normality (s) and during every antigen log one of these values gets incremented ('m' when the cell's K value is positive or 's' when it is zero or lower).

After the global antigen profile has been updated by all the dendritic cells, the 's', 'm' and K values are used to calculate the MCAV (see Equation 1) and $K\alpha$ (see Equation 2) metrics for each antigen type. The results (The Antigen Type's ID, MCAV and $K\alpha$) are printed out on the screen at the end of the run-time.

The DC's statistical data is also being displayed in the Analysis phase. This includes the total number of incarnations and the iterations/incarnations ratio.

8.2 Expansion for Multiple Data Stream Analysis

The design and implementation of the expansion to the DDCA for multiple data stream analysis was one of the key aspects of this project. This section describes the changes made to the original algorithm and explains the decisions taken. Figure 8-2 shows the program flow of the revised model of the Deterministic Dendritic Cell Algorithm.

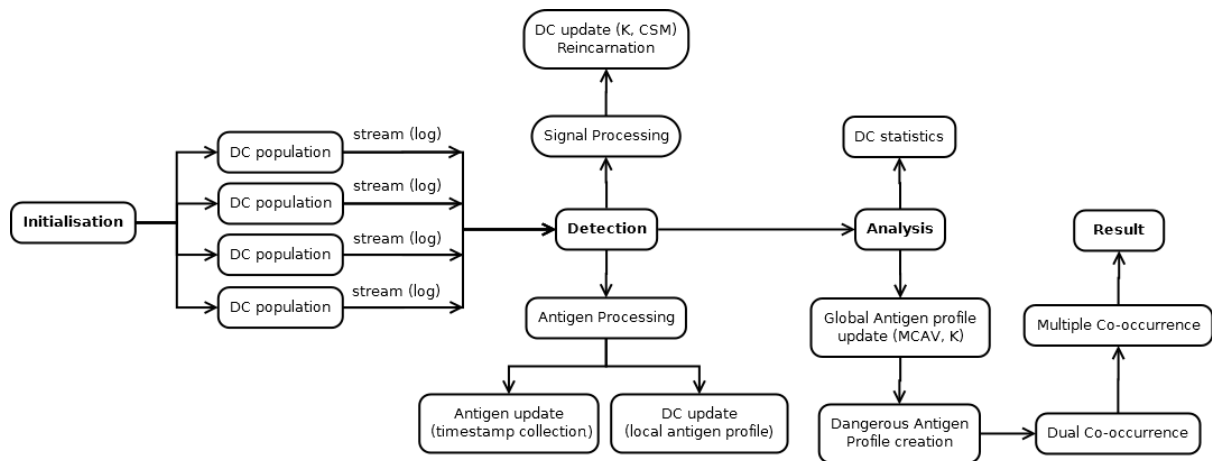


Figure 8-2: modified DDCA program flow (four data streams)

8.2.1 Interface and User Input

In the previous implementation of the DDCA, all of the system's parameters were already hardcoded in the beginning of the algorithm. This model allows the user to select the size of the Dendritic Cell population, set the migration threshold (lifespan limit) as well as enter the number of data streams with the names of the corresponding log files. The current version allows processing of up to four data streams at a single run, however, this number could be increased if necessary (see Maintenance Manual).

8.2.2 Initialisation

After all the parameters and data streams have been selected, the process continues with the initialisation of the Dendritic Cells' populations. As the number of the data streams was now increased, it was decided to initialise several DC populations at once, one for every stream. This prevents the overlapping of data, as every stream's environment should be treated independently. All DC populations are initialised in a similar manner, with an equal number of the dendritic cells and the same migration threshold.

At the beginning of the implementation, every processing stream was being allocated its own global antigen profile. As the program is written in C, this required a complete initialisation and memory allocation for a larger number of profiles than before, right at the start of the program. Moreover, the global antigen profile was now holding antigens that were storing timestamps of their occurrences during run-time. This concept turned out to be very demanding in terms of operational memory and CPU power. Therefore, instead of using a separate profile for each stream, only one global antigen profile has been retained.

This profile was made accessible by all the dendritic cells regardless of the allocated stream. This adjustment has not only made the process more computationally effective but it has also improved the overall accuracy of the algorithm, as the same processes are often observed in various streams and having a single antigen profile from the very beginning allows updating them in a more precise and coherent manner.

8.2.3 Detection

One of the main concerns regarding the Detection phase was whether to use a real-time or an offline analysis. As the main goal of this research was to develop and evaluate new forms of multiple data stream analysis for the Deterministic Dendritic Cell Algorithm, an offline analysis was selected for this purpose. This methodology would provide a better control over the developmental process and more transparency into the run-time environment, making debugging more straightforward and providing more freedom in integrating new forms of analysis.

Another decision was made, based on this solution regarding the concurrency of the processes. Rather than analysing all of the data streams at the same time, the processing of each separate stream is performed sequentially until the final analysis. However, the design and implementation of the current model provides a robust framework for integrating the functionality of processing real-time streaming data.

This version of the DDCA preserves the original processing of the 'signal' data; the CSM and K values are calculated during each input and every DC in the population is updated accordingly.

However, the handling of the 'antigen' input has now been altered. Every input, both 'signal' and 'antigen', has a timestamp reflecting the time of occurrence. In case of an 'antigen' signal, this timestamp is collected and passed to the antigen processing function, which is responsible for storing this data in the global antigen profile. Every antigen instance in this profile can only store a limited number of timestamps (200). This was mainly done to prevent the memory overflow. Moreover, it would simply be impractical to store and analyse such large quantities of data afterwards.

An additional concept was implemented to reduce the number of the timestamps stored. Some of the processes (antigens) tend to run continuously over a long period of time. As we are more interested in capturing multiple occurrences of an antigen rather than a prolonged single one, only those timestamps are collected which vary substantially from the previous ones. This does not imply that a long stream of the same antigen would produce only a single timestamp, but rather that it would be recorded in series.

8.2.4 Analysis

The Analysis phase starts once the data from all the streams have been processed. Afterwards, the DCs in the populations update the global antigen profile in the same manner as in the original DDCA implementation. It is important to mention that there is only one global antigen profile accessible by all the DC populations. Therefore, if the same antigen/process is observed across several streams, the updates from the corresponding dendritic cells affect the same entry in the profile.

For the further analysis, a separate antigen profile is created during the Analysis phase, namely the 'dangerous' antigen profile. It contains antigens that are being considered anomalous; in this case, those are the entries whose MCAV value is higher than zero. This profile is updated during the final calculations of the MCAV and K values. Subsequently, the gathered information is used for the Dual and Multiple Co-occurrence analyses.

The DC's statistical data (for each of the data stream) is also available at this point.

8.2.5 Dual Co-occurrence Analysis

The final stages of the overall data analysis are the Dual and Multiple co-occurrence techniques. Their functioning is based on the information gathered during the Detection and primary Analysis phases.

The Dual co-occurrence method works by iterating through the elements in the newly created 'dangerous' antigen profile and evaluating the correspondence of their collected timestamps. The evaluation of two different antigen types is performed in the following way.

A single antigen structure is able to store up to 200 timestamps. During the analysis, every timestamp of an element is matched against the timestamps of another element. To

produce a successful matching, these timestamps are not required to be exactly the same. However, they should be located within a specified time-window. Every antigen structure holds an additional 2D array that stores the IDs of the antigens that tend to appear together with the given type as well as the total number of their successful matches. The algorithm always checks whether a newfound co-occurrence had previously been recorded, before updating the 2D array of the selected antigen. This feature was designed to avoid duplicates and to have only a single copy of a co-occurrence (instead of both antigens storing the same match in their arrays), which is also required for the implementation of the Multiple Co-occurrence analysis. After all of the anomalous antigens' timelines have been compared, the resulting co-occurrences are displayed on the screen in the following format: Antigen ID – Antigen ID (total number of matches).

8.2.6 Multiple Co-occurrence Analysis

Multiple Co-occurrence uses the results obtained during the Dual Co-occurrence analysis to produce a set of dangerous antigens that are likely to occur together within the same time period (across single or multiple data streams). In terms of a system's security, these findings might indicate possible members of a complex distributed attack. The key source of information for this analysis is the same 'dangerous' antigen profile, although this time the antigens are updated with regard to the dual co-occurrences detected.

A separate profile is created during initialisation that is later used for storing the results of the Multiple Co-occurrence analysis, specifically the groups of antigens that have a likelihood of appearing together. This analysis runs by iterating through the anomalous antigens' 2D array (containing dual co-occurrences) and combining all interconnected types. This method uses a recursive technique. For example, when an antigen 'A' has a co-occurrence with an antigen 'B', the algorithm records this information and then performs the same operation on an antigen 'B' including all its co-occurrences. After this string of interconnected antigens finishes and all the findings are recorded, the program checks the remaining co-occurrences of 'A' in the exact same manner. Therefore, a group of antigens is created when every co-occurrence of 'A' has been processed. Afterwards, the algorithm starts a new group with another element and so on. The results are presented as separate strings of interconnected antigen IDs.

8.2.7 Presenting the Results

After the Initial, Dual and Multiple Co-occurrence analyses have been completed, the results are displayed on the screen and saved in a separate file ("output.txt").

During program runtime, it is possible to see the profiles of every occurred antigen, whether it was considered anomalous or normal. However, the volumes of this information could become extremely large and we are mostly interested in classification of the dangerous elements.

The last output of the program is the same for both the runtime display and the textual representation. The first part is the "Antigen profile", which displays a list of dangerous antigens (IDs) with their corresponding MCAV and K values.

The second part contains the results from the Dual Co-occurrence analysis. It shows pairs of anomalous antigens with the total amount of times their timestamps have matched.

The last part is the Multiple Co-occurrence analysis findings. The groups of interconnected antigens are represented here as strings of IDs.

Chapter 9: Experimentation and Testing

This chapter describes the experiments and testing performed on the expanded Deterministic Dendritic Cell Algorithm. The first section presents the data used for experimentation, explaining its background and format. The following sections describe the experiments performed with regard to the algorithm's performance, scalability, quality of analysis and the effect of various parameter settings.

9.1 Port Scanning Data

The datasets used in these experiments were part of a research project "Articulation and Clarification of the Dendritic Cell Algorithm" [8] by Greensmith, Aickelin and Twycross.

In this concept, port scanning was used as a model of intrusion. Whereas port scan by itself is not an intrusion, it is frequently used for the information gathering by an attacker. This data was collected by observing the activity of outgoing port scans. Even though the analysis of this information by the DDCA would not reveal the attacker immediately, it can be used to identify whether a machine was compromised to send malicious packets over the network or perform additional port scans as part of the information gathering. This could be particularly effective in detecting distributed attacks and botnets.

The data was compiled into ten attack, ten normal and ten control sessions. However, we are mainly interested in the attack sessions, as testing the new forms of analysis requires having several malicious antigens in the datasets. The port scans in the attack sessions were performed by the Nmap scanning tool. The data collected during these sessions was transformed into signal values and written to a file (.log). The danger signals were determined by the number of outgoing network packets per second, because an increase in network traffic could be indicating a suspicious activity. The safe signals were represented by the rate of change of network packets per second. A lower rate would provide a stronger safe signal, as a highly variable rate of sending packets is assumed to indicate the possibility of a malicious behaviour. Figure 9-1 illustrates a single 'signal' input.

```

1147775654.63445 signal 0 100
└────────────────┘ └───┘   |   |
      Timestamp      Identifier Danger Safe
                        signal signal

```

Figure 9-1: 'Signal' Input Format

The first element is the time the signal was observed (displayed in seconds), the second one is the signal identifier and the last two are the danger and safe signal values respectively.

All system calls are recorded during the monitored session. An antigen is represented by the process ID value of a system call that links to the actual process that has made this action. These antigens (process IDs) are collected by the dendritic cells during the DDCA analysis, along with the timestamps that are stored in the global antigen profile. Figure 9-2 shows an example of an 'antigen' input.

```

1147775656.7404 antigen 15993
└────────────────┘ └───┘ └───┘
      Timestamp      Identifier   ID

```

Figure 9-2: 'Antigen' Input Format

As in the case of a 'signal' input, the first element here is the timestamp, the second is an antigen identifier and the last one is the process ID.

9.2 Experiments

9.2.1 Experiment 1 – Analysis Integrity Testing

During the implementation, many core functions of the original Dendritic Cell Algorithm had to be altered to enable the processing of multiple data streams by several DC populations at once. To verify that the analysis of a single data stream still produces the same results, the original DDCA and the modified version of the algorithm were tested on a number of data files and the results compared.

Two datasets were selected for this experiment, namely 's1_norm.log' and 's2_norm.log'. Both of them represent an attack session and contain several malicious processes. The aim of this experiment was to analyse these datasets with the two algorithms and compare the MCAV values of the detected anomalous antigens. Both algorithms were run under the

same parameter settings ('1001' for the number of cells and '10000' for the migration threshold). The hypothesis of this experiment was that the analyses of the original DDCA and the modified model would produce the same results. This would imply that the core functionality of the initial implementation was well preserved in the subsequent version.

9.2.1.1 Results

The results from the analysis of the two datasets are presented in Tables 9-1 and 9-2 respectively.

Algorithm version Antigen ID	Original DDCA	Modified DDCA
16904	0.002417	0.002417
16905	0.007440	0.007440
16954	0.018654	0.018654

Table 9-1: Analysis of the 's1_norm.log'

Algorithm version Antigen ID	Original DDCA	Modified DDCA
8366	0.005329	0.005329
8560	0.023464	0.023464

Table 9-2: Analysis of the 's2_norm.log'

The results show that the algorithms have identified the same antigens (and their MCAV values) after analysing the data from the two data logs. These findings support the hypothesis that the analyses of the original DDCA and the modified model would produce the same results. We can conclude that the core functionality was not distorted during the code alteration.

9.2.2 Experiment 2 – Multiple Data Stream Analysis Testing

The main goal of this experiment was to test the scalability of the algorithm with regard to the number of streams processed at once and measure how this affects the overall performance.

Four different datasets were used for this experiment ('s1_norm.log' to 's4_norm.log'). The performance was measured in time, specifically how long did it take the algorithm to

perform the full analysis. The performance of running up to four streams simultaneously was evaluated. Each stream increment was tested ten times and the average time recorded. The parameters for this experiment were '1001' for the number of cells and '10000' for the migration threshold.

9.2.2.1 Results

Table 9-3 shows the results obtained from the experiment.

Number of Streams	Performance (time in seconds)
1	4.842
2	7.634
3	9.662
4	11.893

Table 9-3: Multiple Data Stream Analysis Performance Testing

The results showed that each additional stream takes three more seconds in average. Further testing revealed that this increase remains constant even after adding extra streams and does not appear to grow progressively.

Initially, each stream was allocated its own global antigen profile. As the profile initialisation is performed in the beginning of the program flow, which requires a substantial amount of operating memory, the CPU resource requirements were increasing rapidly with each additional stream. The maximum number of streams that could have been achieved with that concept was four. However, after reducing the number of global antigen profiles to a single central one, it became possible to expand the number of streams further. The final version of the algorithm is set to process up to four maximum streams, although this could be easily changed if required (see Maintenance Manual). Four streams were sufficient for the further experimentation and these settings were retained for the ease of analysis.

9.2.3 Experiment 3 – Dual Co-occurrence Analysis

This experiment was designed to test the Dual Co-occurrence analysis method. As it was mentioned previously, this technique can search for interconnected antigens within a single stream as well as across multiple data streams. The original data (as it was) was suitable for

detecting interconnected pairs of antigen within a single stream, however, the timelines in the log files were not completely synchronised, which initially did not provide any cross-stream co-occurrences.

For testing purposes, some of the anomalous antigens' timelines were altered so that they would appear at a similar time with the antigens from other streams. These antigens included ID 16904 (co-occurrence with 941), ID 16905 (co-occurrence with 992) and ID 8560 (co-occurrence with 14358).

For this experiment, four different datasets were used ('s1_norm.log' to 's4_norm.log') with 's1_norm.log' and 's2_norm.log' containing the modified timelines. The aim of this test was to verify that the Dual Co-occurrence method is able to identify connections between antigens (regardless of the data stream they appear in). The parameters for this experiment were '1001' for the number of cells and '10000' for the migration threshold.

9.2.3.1 Results

Table 9-4 presents the results from the experiment.

Dual Co-occurrence		
First ID	Second ID	Number of Matches
941	951	1452
941	992	1
941	16904	14
941	16905	23
951	992	17
992	16904	14
992	16905	23
8560	14358	46
14358	14481	556
14358	14482	138
14481	14482	4342
14481	14547	223
14482	14547	762
16904	16905	1408
16905	16954	22

Table 9-4: Results of the Dual Co-occurrence Analysis

As it could have been predicted from the datasets, most of the dual co-occurrences were found within the same data streams. Nevertheless, it can be seen that all of the antigens with the modified timelines were also matched correctly with the antigens from other streams (16904 – 941, 16905 – 992, 8560 – 14358). The changes in the timelines have affected some of the other antigens as well. For example, the antigen 941 has a time period resemblance with the antigen 992, which is reflected in the dual co-occurrence entry, and this similarity has caused a match between the antigens 16904 and 992. A similar case could be observed between the antigens 941 and 16905. Some of the pairs exhibited a higher percentage of matches than the others and these results were verified afterwards by manually evaluating the data logs. To conclude, the modified DDCA has proven to be successful in identifying connections between pairs of antigens both within a single and across multiple data streams.

9.2.4 Experiment 4 – Multiple Co-occurrence Analysis

The aim of this test was to evaluate the performance of the Multiple Co-occurrence analysis. This method produces strings of interconnected elements based on the results from the Dual Co-occurrence analysis. The datasets from the Experiment 3 were reused to perform this test and the parameters were set to '1001' for the number of cells and '10000' for the migration threshold.

As this technique is essentially an extension to the Dual Co-occurrence method, the accuracy of the algorithm could be reviewed manually by referring to the results of the previous experiment.

9.2.4.1 Results

The resulting antigen strings are shown in Table 9-5.

String number	Interconnected Antigens
1	941 951 992 16904 16905 16954
2	8560 14358 14481 14482 14547

Table 9-5: Results of the Multiple Co-occurrence Analysis

The results obtained in this experiment were compared to the manually created strings of antigens based on the findings from the Dual Co-occurrence analysis. The two sets of strings were identical, which demonstrates the correctness of the given implementation.

9.2.5 Experiment 5 – Number of Cells in the DC Population

This experiment was performed to test the sensitivity of the algorithm to various sizes of the DC population. As the Dual and Multiple Co-occurrence analyses' performance is mainly based on the calculations of the MCAV values during the primary analysis stage, this experiment aims to investigate how changing the number of cells in the population could affect these calculations and the number of antigens identified. Testing the sensitivity of the parameters was part of the future work mentioned in the "Articulation and Clarification of the Dendritic Cell Algorithm" [8]; this was one of the main reasons for conducting this and the following experiments. The tests were performed on the same four datasets ('s1_norm.log' to 's4_norm.log') with four different sizes of the DC population ('1000', '800', '600', '400'). A size of 1000 dendritic cells was taken as the benchmark for the experimental

settings and results analysis, as it has been used for this data's analysis in the original research ('1001' cells was the exact number, but to keep the distribution more uniform and to test if small discrepancy will affect the analysis, '1000' was used). The results in that research showed that the algorithm with those settings was able to correctly identify all the dangerous antigens without producing any false positives. The aim of this experiment was then set to investigate whether lowering the number of cells in the population could produce the same results, while preserving some of the CPU resources. The migration threshold was retained constant throughout the experiment ('10000').

9.2.5.1 Results

Table 9-6 shows the calculated MCAV values under various DC population sizes.

Size of Population '1000'		Size of Population '800'		Size of Population '600'		Size of Population '400'	
Antigen ID	MCAV	Antigen ID	MCAV	Antigen ID	MCAV	Antigen ID	MCAV
941	0.096841	941	0.104291	941	0.101311	941	0.102801
951	0.048726	951	0.035982	951	0.032234	951	0.034483
992	0.139417	992	0.143341	992	0.142019	992	0.137764
8366	0.006217	8366	0.012433	8366	0.008881	8366	0.009769
8560	0.023126	8560	0.023295	8560	0.022113	8560	0.024139
14358	0.009551	14358	0.009551	14358	0.000955	14358	0.006686
14481	0.085049	14481	0.091614	14481	0.089824	14481	0.087138
14482	0.015284	14482	0.005822	14482	0.007278	14482	0.026929
14547	0.103442	14547	0.101880	14547	0.102314	14547	0.102227
16904	0.005641	16904	0.006446	16904	0.004835	16904	0.004029
16905	0.009673	16905	0.004464	16905	0.008185	16905	0.004464
16954	0.017885	16954	0.020000	16954	0.019615	16954	0.019038

Table 9-6: DC Population Size Testing Results

The results showed that 1000 cells have identified the same antigens as 1001; however, the MCAV values were generally slightly lower in the first case. It can be seen that lowering the size of the DC population from 1000 to 800 has a varied effect on the MCAV value: some of the values increase, some stay at the same level and some decrease. However, decreasing the number of cells further causes a general decline of the MCAV value (with some exceptions).

Another interesting finding was that starting from the size of 800 cells and lower, there was a gradual increase in the number of dangerous antigens identified compared to 1000 cells (800 cells: ID 875, ID 954, ID 959, ID 14485, ID 14486; 600 cells: ID 875, ID 877, ID 954, ID 959, ID 14380, ID 14485, ID 14486; 400 cells: ID 875, ID 877, ID 954, ID 959, ID 14380, ID 14485, ID 14486, ID 16915). Smaller sizes of a DC population seem to identify more antigens as dangerous while generally allocating a lesser MCAV value. In this case, the extra antigens identified were false positives, however, for each new dataset it is important to verify the results that were obtained with various DC population sizes. Setting this parameter too high might lead to the DDCA missing some of the dangerous antigens while having it too small might increase the rate of false positives. The findings in this experiment were contrary to the observations in the paper “Dendritic Cells Algorithm and Its Application to Nmap Portscan Detection” [7] which stated that the algorithm was not particularly sensitive to the parameter settings’ changes.

9.2.6 Experiment 6 – Migration Threshold

The aim of this experiment was to test how changing the migration threshold affects the overall data analysis. The design of this test is similar to that of the previous experiment, except that the migration threshold has now become the new independent variable instead of the number of cells in the population. The test runs were performed on four datasets (‘s1_norm.log’ to ‘s4_norm.log’) with four different migration threshold settings (‘10000’, ‘8000’, ‘6000’, ‘4000’). The number of cells in the population was now kept constant through the whole experiment (‘1000’). This time a migration threshold of ‘10000’ was selected as the benchmark for the experiment settings and results’ analysis (based on the settings in the original implementation).

9.2.6.1 Results

Table 9-7 displays the calculated MCAV values under various migration threshold settings.

Migration Threshold '10000'		Migration Threshold '8000'		Migration Threshold '6000'		Migration Threshold '4000'	
Antigen ID	MCAV	Antigen ID	MCAV	Antigen ID	MCAV	Antigen ID	MCAV
941	0.096841	941	0.085816	941	0.070620	941	0.065554
951	0.048726	951	0.038231	951	0.013493	951	0.013493
992	0.139417	992	0.122150	992	0.107031	992	0.112153
8366	0.006217	8366	0.006217	8366	0.007105	8366	0.006217
8560	0.023126	8560	0.022451	8560	0.023801	8560	0.024139
14358	0.009551	14358	0.006686	14358	0.027698	14358	0.020057
14481	0.085049	14481	0.100269	14481	0.140555	14481	0.201432
14482	0.015284	14482	0.022562	14482	0.079330	14482	0.066230
14547	0.103442	14547	0.125537	14547	0.170248	14547	0.193992
16904	0.005641	16904	0.004835	16904	0.004835	16904	0.008058
16905	0.009673	16905	0.006696	16905	0.005952	16905	0.005952
16954	0.017885	16954	0.016731	16954	0.017500	16954	0.019615

Table 9-7: Migration Threshold Testing Results

It is possible to tell from the results that changing the migration threshold has a mixed effect on the MCAV outcome: almost half of the values increased while the others decreased and no evident trends were observed in these findings.

Furthermore, the effect of changing the migration threshold was not as significant as the effect of changing the number of cells in the DC population. This notion was supported by the fact that only two more anomalous antigen IDs (false positives) were identified after

lowering the migration threshold value (ID 14372 and ID 14380) whereas in the previous experiment eight more anomalous antigens were recorded. The results obtained in this experiment display a closer resemblance to those described in the paper “Dendritic Cells Algorithm and Its Application to Nmap Portscan Detection” [7]. However, the MCAV difference between the extreme values (10000 and 4000) is still relatively high.

Chapter 10: SEPA River Data

One of the secondary goals was to test the modified algorithm on a new dataset from another domain. The new data was provided by Dr. Yaji Sripada; it contains the Scottish river levels gathered by the Scottish Environment Protection Agency [9]. This concept was mostly theoretical as there was not enough insight into the specifics of threats and dangers that could be identified using this type of analysis and which circumstances and events best represent these situations. However, it was a good experience of applying the DDCA to a novel dataset, testing the new co-occurrence methods and incorporating some of the concepts from other related works. This description could serve as a possible guide for modifying the algorithm for new data processing. This chapter describes the new dataset, outlines the problem domain, explains the necessary changes made to the DDCA and presents the experiments performed.

10.1 Data Description

SEPA monitors water levels at more than 300 sites throughout Scotland. The data provided displays the water level (height of water in the river) measured in meters at different river stations. The data file is structured into blocks of data, each block corresponding to one measuring station along the river. The first block, for example, corresponds to a measuring station in Alford.

Each block of data has a header that contains metadata and then followed by a stream of real data. In this concept the metadata is used for extracting the name of the river station, which in this case represents the name of the antigen. Figure 10-1 illustrates the first line of a data block that indicates which station the measures were taken from.

```
## Exported ZRXP Block for Alford.SG.ir.O
Antigen Identifier      Antigen Name
```

Figure 10-1: SEPA 'Antigen' Input Format

The signals in this model are represented by the real data. A single signal input has three columns. The first column is the timestamp, the second is the river height in meters and the third column is a quality flag (ignored in this case due to being context specific). This signal input format is shown in Figure 10-2.

20130303000000 1.165 0
Timestamp Signal/Height

Figure 10-2: SEPA 'Signal' Input Format

10.2 Problem Domain

As the data does not provide any specific information regarding the possible danger or threats, river height was chosen as the basis for the signal analysis. What the algorithm tries to indicate in this case, are the stations that are likely to become a cause of the river overflow or flooding. Two factors are taken into account when calculating this possibility. Firstly, if a river height is above a certain threshold, the corresponding river station has a higher chance of being classified as dangerous. Secondly, rapid height changes within a single station are also considered in this model as the likely indicators of a dangerous situation.

Rapid height changes were also used for calculating the possible connections between various river stations. For example, if two river stations have a series of rapid height changes within the same time window, these co-occurrences are recorded and the higher the number of these co-occurrences the more likely is the connection between the two river stations. The only problem with this concept is not knowing how far these two stations are from each other. The distance between the two might cause a time delay that is hard to account for without knowing this information. However, this could become a part of the future work, together with verifying what aspects are more likely to cause flooding in these areas.

10.3 The DDCA Adaptation

The DDCA version used for port scanning data could not be applied to analysing the SEPA river data due to the differences in the format and content. Therefore, a new version of the modified DDCA was made for the analysis of the new datasets. The core functionalities remained the same, but some of the design principles had to be altered.

10.3.1 Signals and Antigens

The concepts of signals and antigens were slightly redefined in this model. The antigens were now represented by the river stations. As the data blocks, and therefore the river station names, were now presented consequently rather than scattered around across the whole dataset (as in the case of port scanning data), each antigen was only presented once through the whole dataset followed by a series of real data containing the river height at that station over a period of time.

For this reason, a concept of the antigen multiplier, described in the paper “A Sense of ‘Danger’ for Windows Processes” [5], was used in this model to increase the number of Dendritic Cells that would process any given antigen, thus increasing the quality of the analysis. Although the technique described in the paper was performed by copying each antigen’s occurrence several times, it was decided to allocate fifty additional dendritic cells for the processing of each antigen occurrence instead. This has produced the same effect; the only difference was the implementation. In addition to saving the name, each new river station (antigen) is allocated an ID number for the ease of analysis.

The signals had to be redefined as well, as there were no safe and danger values in the new data. The concept of K and CSM output was retained, only this time these values were calculated based on the water level and the occurrence of rapid height changes. For example, if the river height was above certain threshold or there were any rapid height changes, the K value would be positive (indicating danger), whereas in the normal circumstances the K value would be zero or negative. An observation from the paper “DCA for Bot Detection” was taken into account when the K value for the safe signals was being selected. The authors stated that higher safe signal values tended to reduce the potential false positives. Considering the fact that the current data has a very high rate of signal inputs (especially safe) compared to other datasets, K value of a medium value was selected in order to keep a gradual update rate. The CSM value was set higher in case of a dangerous situation and lower otherwise. These values are used to update the dendritic cells in the populations as it was done in the previous implementations.

As there was now only one occurrence of each antigen, the timestamp collection was done during signal processing. The ID number of every new occurring antigen is saved in a global

parameter that is used by the algorithm to correctly allocate the timestamp to the given antigen's profile, which is only done in the case of rapid height changes. The concept of time windows is also used here to reduce the number of collected timestamps.

10.3.2 Analysis

The calculation of MCAV and K values are done in the same way as in the previous implementations. However, during the calculations of these values two additional antigen profiles are created. The first one is the 'dangerous' antigen profile containing antigens whose MCAV value is larger than zero. The other one is created specifically for storing river stations (antigens) where rapid height changes have been detected. This profile is subsequently used for Dual and Multiple Co-occurrence analyses.

10.3.3 Dual and Multiple Co-occurrence Analyses

The Dual and Multiple Co-occurrence analyses have not been changed since the previous implementation. The only difference is that they are now comparing antigens for the co-occurrence of rapid height changes within the same period. For this purpose, the newly created profile is used that contains all the river stations where rapid height changes have been recorded.

10.4 Experimentation and Testing

Every SEPA data file contains information regarding the same river stations during different periods. Therefore, the experiments were performed using only one stream (data file). It would have been reasonable to use several streams if they contained information about different river stations within the same time. The new algorithm was tested on performing the MCAV calculations as well as Dual and Multiple Co-occurrence analyses.

10.4.1 MCAV calculation

The aim of this experiment was to evaluate how the modified DDCA would perform on the new dataset, how many antigens would be identified and what MCAV values would they receive. The new concept works in a way such that a station with a high K value is likely to affect the nearby station values as well, until the environment is normalised again. This should produce a series of river stations with the most 'dangerous' elements being in the beginning or in the middle of these series. The 'sepa1.log' data file was used in this

experiment. The parameters were '1000' for the number of cells and '10000' for the migration threshold.

10.4.1.1 Results

Table 10-1 presents the results obtained from the experiment.

Series of Antigens (ID)	Most Dangerous Element (MCAV)
38	ID 38 (0.078431)
58 - 74	ID 61 (0.039216)
157 - 173	ID 166 (0.313726)
212 - 221	ID 216 (0.215686)
289 – 291	ID 289 (0.156863)
312 - 330	ID 330 (0.901961)

Table 10-1: SEPA Data MCAV Experiment Results

The results from the experiment show that most of the antigens were identified in series with the most 'dangerous' antigens usually located in the beginning or in the middle of a sequence. The only exception was the last sequence, with antigen 330 being the most anomalous. These findings were verified manually against the log data and most of the antigens identified either had a high water level or a series of rapid height changes. Some of the antigens were simply affected by the nearby 'dangerous' elements. River station under the ID 330 had an extremely high water level together with rapid river height changes, which has significantly increased its MCAV value. Overall, the algorithm was effective in identifying these river qualities and assigning appropriate MCAV values.

10.4.2 Dual Co-occurrence Analysis

This experiment was performed on the same dataset ('sepa1.log') to test the Dual Co-occurrence analysis. As opposed to the MCAV calculation, this method only takes into account rapid height changes that happen within the same time window across different stations. One station's timeline was artificially altered to check if the algorithm would be able to detect this co-occurrence (it was verified that these stations did not have any previous co-occurrences). The 'Newcastlet.SG.ir.O' (ID 100) was set to have several co-occurrences with the station 'Park.Q.15' (ID 328). The parameters were '1000' for the number of cells and '10000' for the migration threshold.

10.4.2.1 Results

The results from the experiment are shown in Table 10-2.

Station (ID)	Dual Co-occurrence with (ID – number of times)
ID 16	ID 29 (4), ID 38 (1), ID 163 (3), ID 217 (5), ID 289 (4), ID 328 (4)
ID 29	ID 38 (1), ID 163 (3), ID 217 (5), ID 289 (5), ID 328 (11)
ID 38	ID 163 (4), ID 217 (5), ID 289 (5), ID 328 (15)
ID 100	ID 328 (5)
ID 163	ID 168 (4), ID 217 (12), ID 289 (6), ID 328 (30)
ID 168	ID 217 (4), ID 328 (12)
ID 217	ID 289 (9), ID 328 (39)
ID 289	ID 328 (24)

Table 10-2: SEPA Data Dual Co-occurrence Experiment Results

Firstly, it can be seen that the station ID 100 (with the altered timeline) was correctly linked to the station ID 328. Secondly, the stations that are linked with other stations the most (ID 289, ID 328) always appear to have a relatively large MCAV value (0.156863 and 0.647059 respectively). To conclude, the Dual Co-occurrence technique has proven to be working correctly. However, it would be important to tune the parameters of the algorithm in accordance with the evidence regarding the links between these stations in the future.

10.4.3 Multiple Co-occurrence Analysis

The goal of the final experiment was to test the Multiple Co-occurrence analysis method. By examining the results from the Dual Co-occurrence analysis, it can be concluded that all of these stations are interconnected in one way or another. Therefore, the expected outcome of running this experiment on the same dataset would be a string of nine interconnected stations/antigens. The parameters were the same as well: '1000' for the number of cells and '10000' for the migration threshold.

10.4.3.1 Results

After running the test only a single string of antigens was produced, specifically:

ID 16 (LutherBrid.SG.ir.O) ID 29 (Apigill.SG.ir.O) ID 38 (Corpach.SG.ir.O) ID 163 (RenfrewTid.SG.ir.P) ID 168 (TarbertTid.SG.ir.P) ID 217 (Clachnahar.SG.ir.P) ID 289 (ArbroathTi.SG.ir.P) ID 328 (Park.Q.15) ID 100 (Newcastlet.SG.ir.O)

This string contains all the interconnected river stations from the previous Dual Co-occurrence analysis. This exhibits the correct behaviour of the algorithm, which supports the results gained in the Experiment 4 of the port scanning data.

Chapter 11: Summary and Conclusion

This chapter summarises and discusses the tasks accomplished, providing a critical evaluation of the work that was done and suggesting future development.

11.1 Summary

Looking back at the project goals and requirements, it is possible to say that almost all of them were successfully achieved.

- **Understanding the principles behind the Deterministic Dendritic Cell Algorithm and designing an appropriate expansion to account for multiple data stream processing.**

The Deterministic Dendritic Cell Algorithm, its biological background and algorithmic adaptation were thoroughly studied and described in theory as well as practical applications. This was an excellent introduction to the field of Artificial Immune Systems and showed the potential of the systems in the information security domain. This experience provided a good basis for expanding the existing Deterministic Dendritic Cell Algorithm. Although, the multiple data stream processing was implemented as a simulation rather than a real-time parallel analysis, it has provided the necessary framework for the new forms of analysis. Furthermore, the new design and architecture laid the groundwork for the implementation of real-time analysis of live streaming data.

- **Extending the analysis to find co-occurrences between separate data streams**

The use of timestamps was successfully incorporated in the modified Dendritic Cell Algorithm. This led to the development of the new forms of analysis, namely the Dual and Multiple Co-occurrence analyses that are able to detect simultaneous runs of malicious processes occurring on different machines (streams). This could be particularly effective for detecting botnet type attacks. When a network of subverted machines receives a command from the central attacker, this could lead to a series of malicious actions that occur at the same time on a number of different machines. The original DDCA's capability of detecting malicious processes is

supplemented in this case with the functionality to detect these co-occurrences across a range of streams.

- **Testing the resulting application on a dataset from a real-world domain**

The modified algorithm was initially tested on the port scanning data that was used previously by Julie Greensmith in her research [8]. Firstly, it was verified that the core functionality was well preserved in the modified algorithm. Secondly, the new Dual and Multiple Co-occurrence analyses were tested. The results from the experiment showed that the modified algorithm was working as intended, having preserved the original functionality as well as displaying co-occurrences accurately. The Multiple Co-occurrence analysis' performance was evaluated and verified based on the results from the Dual Co-occurrence analysis.

- **Perform experiments on the dataset to assess the effectiveness of the modified algorithm while making necessary changes to the parameter settings for a better optimisation and performance**

The algorithm's performance was tested as well as its sensitivity to some of the parameter changes. Some of the results were contrary to earlier findings, showing that the DDCA was sensitive to the changes made to the number of cells in the population. A higher number of cells tends to produce more accurate results, whereas setting a low number might result in identifying a number of false positives. It is always important to test the algorithm against real evidence to set the parameters right, as setting the number of cells too high might result in some of the 'dangerous' antigens being omitted. The sensitivity of the algorithm to the changes of the migration threshold was not as strong as with the number of cells in the population.

- **Obtaining a second dataset from another domain and testing the modified algorithm, after experimentation on the initial dataset (secondary task)**

SEPA river data provided by Dr. Yaji Sripada was chosen for the second dataset. It contains water level measured in meters at different river station across Scotland. As it was mentioned earlier, this was a theoretical concept, mostly aimed at testing the

modified Deterministic Dendritic Cell Algorithm and the new forms of analysis. A completely new dataset required a further modification of the DDCA, and some of the concepts had to be redefined. A couple of new features from the related works (Antigen Multiplier, safety signal values) were successfully applied to the new model. The calculations of the MCAV values, Dual and Multiple Co-occurrence analyses were tested on the new data and the results were satisfactory. However, more evidence is needed regarding the real threats that are associated with the river height. Nonetheless, this was a valuable experience and this concept could be later used as a guideline for applying the DDCA to novel datasets.

The project meets almost all of the primary and secondary requirements. Some of the secondary goals were not accomplished, namely the development of a suitable GUI for the system. More time was invested into developing a second version of the modified DDCA for the SEPA river data analysis. Moreover, the idea of creating a GUI for the DDCA was later reconsidered, as the number of parameters that the user needs to control is not that high. A simple terminal interface was sufficient for this task.

11.2 Future Development

There are many possible ways for developing the project further. Some of these ideas concern the development of the Deterministic Dendritic Cell Algorithm in general; some could be applied to the specific domain that the algorithm was designed to work with.

- **Improve the algorithm to perform real-time analysis of live streaming data**

The design and architecture of the modified DDCA laid a solid framework for extending the algorithm for performing real-time analysis, rather than collecting the data at first and then using it for the analysis. A possible approach for this is explained in the paper “Integrating Real-Time Analysis With The Dendritic Cell Algorithm Through Segmentation” [10] by Feng Gu, Julie Greensmith and Uwe Aickelin. The paper describes a concept of using segmentation for merging the Detection and Analysis phases together to perform the analysis in the process of collecting live streaming data. Although this analysis is not completely real-time yet,

as the segmentation method still entails a certain time delay, this is a good starting point for developing the methodology further.

- **Creating graphical representation of the results**

Another option for further development is the creation of graphical representation of the analysis results. The difficulty here is that some of the datasets return a high volume of various output (e.g. anomalous antigens, co-occurrences, etc.) and displaying all this information on a single chart might be inconvenient. Nonetheless, it is a good idea to display at least the most basic and important findings to give a quick overview of the results.

- **Improving the SEPA river data analysis**

The DDCA for SEPA river data analysis could be improved by obtaining evidence regarding the danger and threats that are associated with the river height level and rapid height changes. Incorporating information regarding the location of different river station could also significantly improve the analysis.

- **Adding new forms of analysis that could be achieved by analysing timestamps**

It is possible to incorporate more forms of statistical analysis into the existing algorithm by using the newly developed framework and the collection of timestamps.

11.3 Conclusion

In conclusion, this project could be rated as successful. This research was an enlightening experience into the world of Artificial Immune Systems, especially the principles of the Deterministic Dendritic Cell Algorithm and how it could be applied in the real-world domain.

Most of the primary and secondary project goals were achieved. The modified DDCA incorporated new principles of multiple data stream processing and new forms of analysis, namely the Dual and Multiple Co-occurrence analysis techniques. The modified versions of the algorithm were tested on port scanning data as well as SEPA river data and the results were satisfactory.

Many challenges were faced during the design and implementation of the new algorithm; many alternative paths considered. Nonetheless, this research project has provided many valuable insights into the principles and application of the DDCA. The results gained from the experiments highlighted many important features of the algorithm, such as its sensitivity to the parameter settings.

There are many paths open for the future development. The modified versions of the algorithm could be used as a valuable supplement to the original DDCA analysis or even applied to a wider range of domains.

Bibliography

- [1] C. Musselle, "Rethinking Concepts of the Dendritic Cell Algorithm for Multiple Data Stream Analysis," Department of Computer Science, University of Bristol, UK, 2012.
- [2] J. Greensmith, U. Aickelin and J. Twycross, "Articulation and Clarification of the Dendritic Cell Algorithm," CS&IT, University of Nottingham, UK, NG8 1BB, 2006.
- [3] J. Greensmith and U. Aickelin, "The Deterministic Dendritic Cell Algorithm," School of Computer Science, University of Nottingham, UK, NG8 1BB, 2008.
- [4] F. Gu, J. Greensmith and J. Aickelin, "Theoretical Formulation and Analysis of the Deterministic Dendritic Cell Algorithm," School of Computer Science, University of Nottingham, NG8 1BB, UK, 2013.
- [5] S. Manzoor, M. Z. Shafiq, S. M. Tabish and M. Farooq, "A Sense of 'Danger' for Windows Processes," FAST National University of Computer & Emerging Sciences (NUCES), Islamabad, 44000, Pakistan.
- [6] Y. Al-Hammadi, U. Aickelin and J. Greensmith, "DCA for Bot Detection," Department of Computer Science, The University of Nottingham, Jubilee Campus, Nottingham NG8 1BB, UK.
- [7] F. Xianjin and S. Danjie, "Dendritic Cells Algorithm and its Application to Nmap Portscan Detection," School of Computer Science, Anhui University of Science and Technology, Huainan, China, 2012.
- [8] F. Gu, J. Greensmith and U. Aickelin, "Integrating Real-Time Analysis With The Dendritic Cell Algorithm Through Segmentation," School of Computer Science, University of Nottingham, Nottingham, UK, NG8 1BB, 2009.
- [9] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes*, Lulu Enterprises, 2011.

[10] S. E. P. Agency, "Scottish River levels," [Online]. Available:
http://www.sepa.org.uk/water/river_levels.aspx. [Accessed 10 May 2013].

Appendix A: User Manual

The executable versions of the program (together with the data files) could be found in the 'executable' folder in project submission.

There are two different variations of the DDCA algorithm: ddcaPORT and ddcaSEPA.

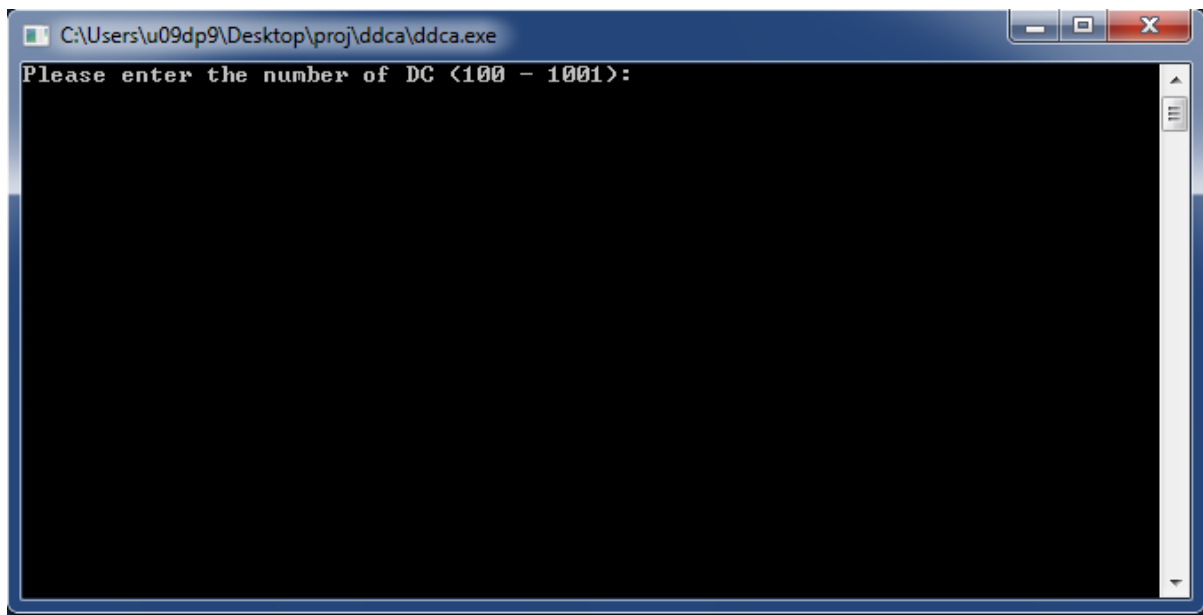
Use the ddcaPORT for the analysis of port scanning data.

Use the ddcaSEPA for the analysis of SEPA river data.

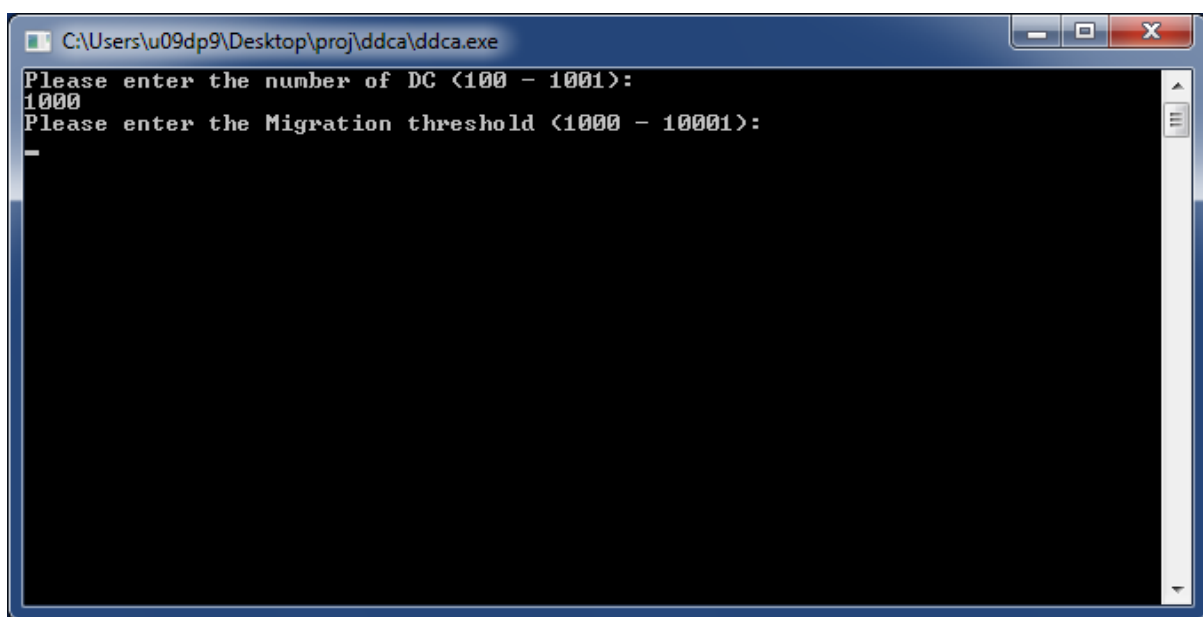
The two programs are run in the exact same way, except for some of the minor differences (data file names, output variations, etc.) which is explained in this manual.

- To start the program on Windows, open the folder 'ddcaPORT' ('ddcaSEPA') that contains the executable file 'ddcaPORTwin.exe' ('ddcaSEPAwin.exe') and the data files that you want to analyse ('s1_norm.log' to 's4_norm.log' for the port scanning data or 'sepa1.log' to 'sepa3.log' for the SEPA river data) and simply double click on the executable file.
- To run the program on a Unix/Linux system, open a terminal in the folder 'ddcaPORT' ('ddcaSEPA'). Make sure that the compiled C program 'ddcaPORTunix' ('ddcaSEPAunix') and the data files for the analysis are in the same folder ('s1_norm.log' to 's4_norm.log' for the port scanning data or 'sepa1.log' to 'sepa3.log' for the SEPA river data). To start the program, simply type in the terminal: `./ddcaPORTunix` (or `./ddcaSEPAunix`) without the double quotes.

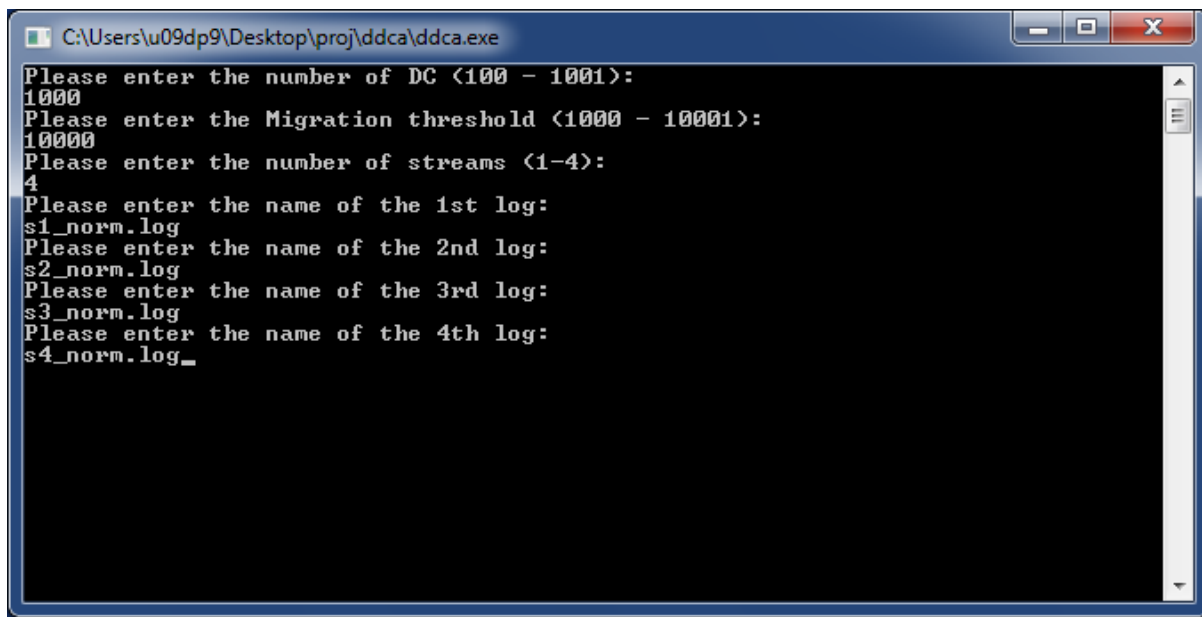
After starting the program you are asked to choose the number of DC (dendritic cells) in the population (a number from 100 to 1001):



After entering the number of DC for the population and pressing enter, you are asked to enter the Migration Threshold value (maximum DC lifespan) and press enter:



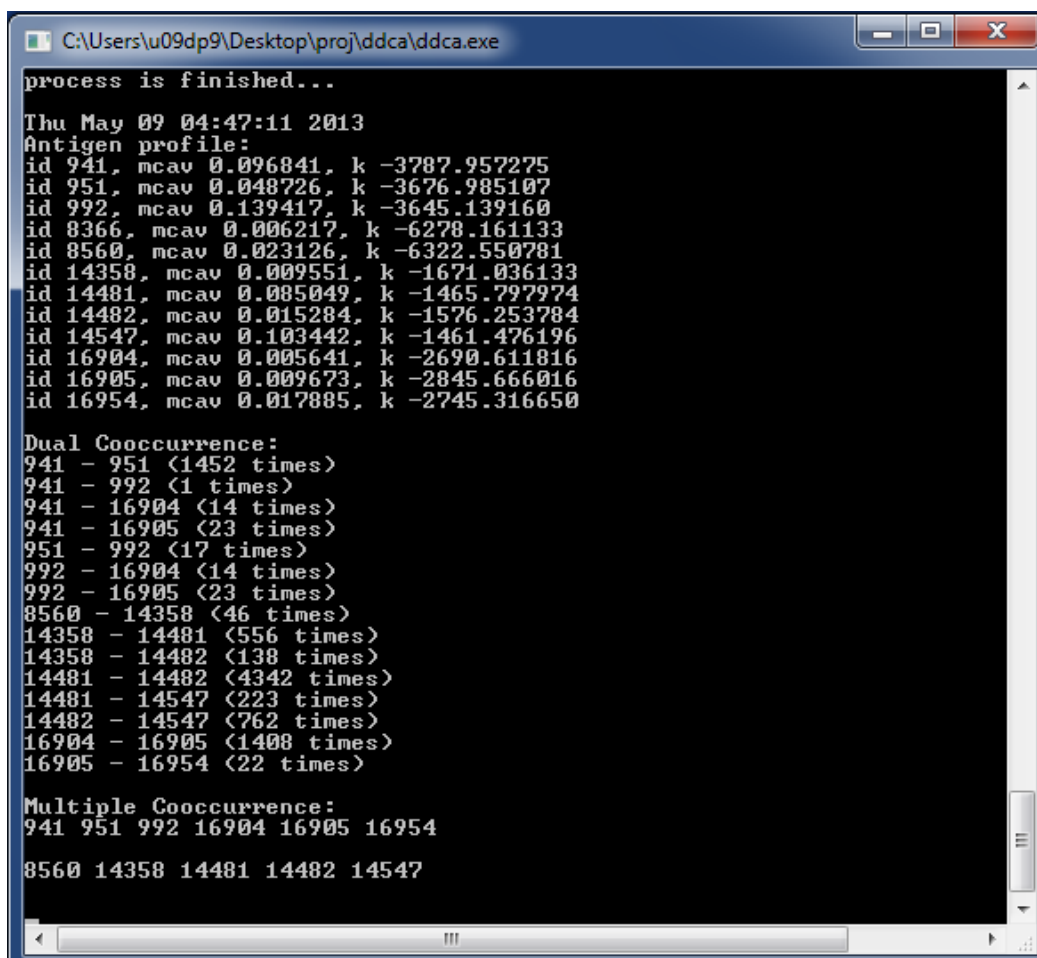
Finally, you need to enter the number of streams (data logs) you want to process and type in their names afterwards:



```
C:\Users\u09dp9\Desktop\proj\ddca\ddca.exe
Please enter the number of DC (100 - 1001):
1000
Please enter the Migration threshold (1000 - 10001):
10000
Please enter the number of streams (1-4):
4
Please enter the name of the 1st log:
s1_norm.log
Please enter the name of the 2nd log:
s2_norm.log
Please enter the name of the 3rd log:
s3_norm.log
Please enter the name of the 4th log:
s4_norm.log_
```

After this the analysis starts. It is usually performed fairly quickly, with the analysis of SEPA river data taking slightly longer.

In the end of the analysis you are presented with the final results:



```
C:\Users\u09dp9\Desktop\proj\ddca\ddca.exe
process is finished...
Thu May 09 04:47:11 2013
Antigen profile:
id 941, mcau 0.096841, k -3787.957275
id 951, mcau 0.048726, k -3676.985107
id 992, mcau 0.139417, k -3645.139160
id 8366, mcau 0.006217, k -6278.161133
id 8560, mcau 0.023126, k -6322.550781
id 14358, mcau 0.009551, k -1671.036133
id 14481, mcau 0.085049, k -1465.797974
id 14482, mcau 0.015284, k -1576.253784
id 14547, mcau 0.103442, k -1461.476196
id 16904, mcau 0.005641, k -2690.611816
id 16905, mcau 0.009673, k -2845.666016
id 16954, mcau 0.017885, k -2745.316650

Dual Cooccurrence:
941 - 951 (1452 times)
941 - 992 (1 times)
941 - 16904 (14 times)
941 - 16905 (23 times)
951 - 992 (17 times)
992 - 16904 (14 times)
992 - 16905 (23 times)
8560 - 14358 (46 times)
14358 - 14481 (556 times)
14358 - 14482 (138 times)
14481 - 14482 (4342 times)
14481 - 14547 (223 times)
14482 - 14547 (762 times)
16904 - 16905 (1408 times)
16905 - 16954 (22 times)

Multiple Cooccurrence:
941 951 992 16904 16905 16954

8560 14358 14481 14482 14547
```

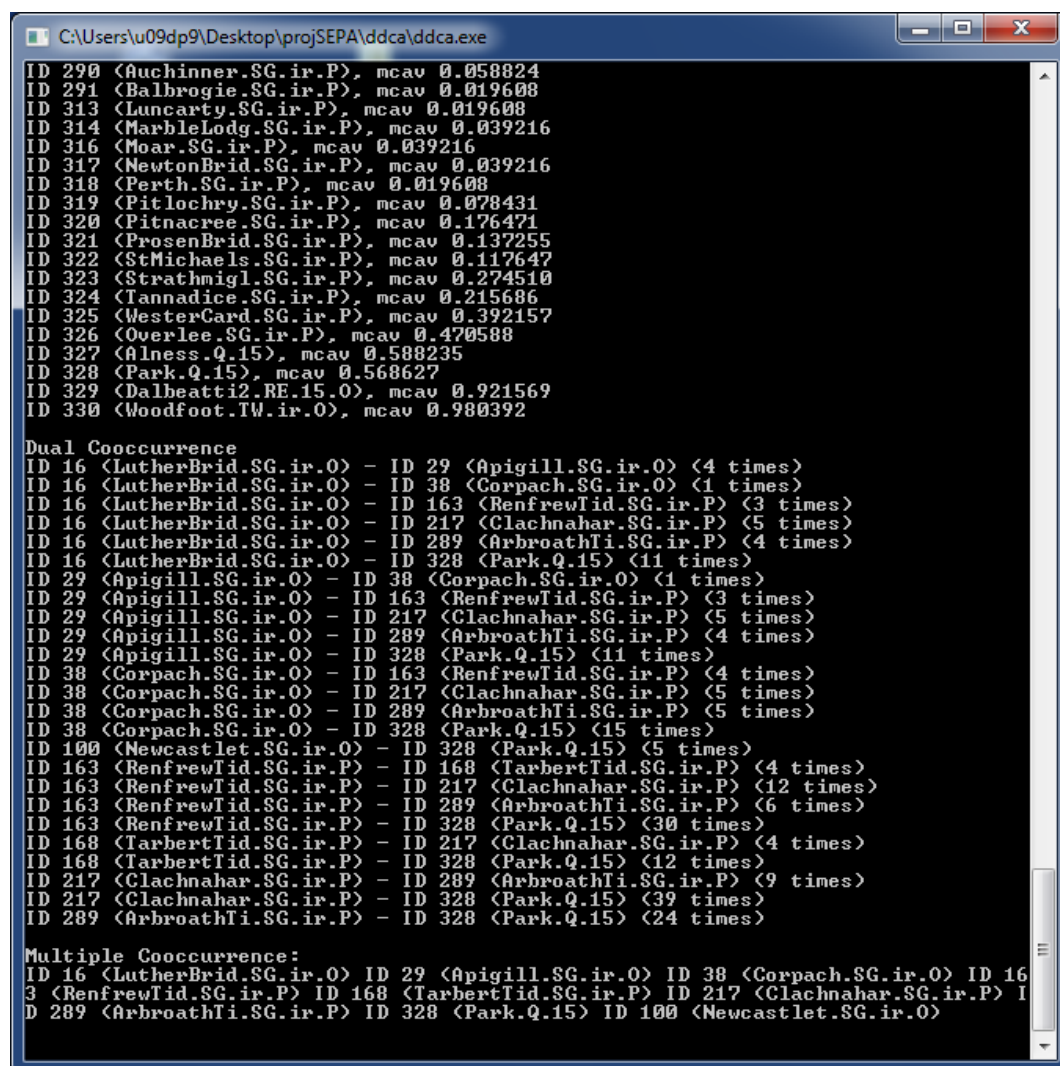
The first part is the Antigen profile. It presents the IDs of processes that are considered 'dangerous', together with their K and MCAV values.

The second part is the Dual Co-occurrence Analysis results, containing pairs of 'dangerous antigens' that occur within the same time windows. The output presents the two IDs and the number of times their timestamps have matched.

The final part is the Multiple Co-occurrence analysis results, which shows groups of interconnected antigens.

You can also find these results in a file called "output.txt" that is created (or gets updated) every time you run an analysis.

The results of the SEPA river data analysis are presented in a similar manner, except the names of the river stations are also included in the report:



```
C:\Users\u09dp9\Desktop\projSEPA\ddca\ddca.exe
ID 290 <Auchinner.SG.ir.P>, mcau 0.058824
ID 291 <Balbrogie.SG.ir.P>, mcau 0.019608
ID 313 <Luncarty.SG.ir.P>, mcau 0.019608
ID 314 <MarbleLodg.SG.ir.P>, mcau 0.039216
ID 316 <Moar.SG.ir.P>, mcau 0.039216
ID 317 <NewtonBrid.SG.ir.P>, mcau 0.039216
ID 318 <Perth.SG.ir.P>, mcau 0.019608
ID 319 <Pitlochry.SG.ir.P>, mcau 0.078431
ID 320 <Pitnacree.SG.ir.P>, mcau 0.176471
ID 321 <ProsenBrid.SG.ir.P>, mcau 0.137255
ID 322 <StMichaels.SG.ir.P>, mcau 0.117647
ID 323 <Strathmig1.SG.ir.P>, mcau 0.274510
ID 324 <Tannadice.SG.ir.P>, mcau 0.215686
ID 325 <WesterCard.SG.ir.P>, mcau 0.392157
ID 326 <Overlee.SG.ir.P>, mcau 0.470588
ID 327 <Alness.Q.15>, mcau 0.588235
ID 328 <Park.Q.15>, mcau 0.568627
ID 329 <Dalbeatt12.RE.15.O>, mcau 0.921569
ID 330 <Woodfoot.IW.ir.O>, mcau 0.980392

Dual Cooccurrence
ID 16 <LutherBrid.SG.ir.O> - ID 29 <Apigill.SG.ir.O> <4 times>
ID 16 <LutherBrid.SG.ir.O> - ID 38 <Corpach.SG.ir.O> <1 times>
ID 16 <LutherBrid.SG.ir.O> - ID 163 <RenfrewTid.SG.ir.P> <3 times>
ID 16 <LutherBrid.SG.ir.O> - ID 217 <Clachnahar.SG.ir.P> <5 times>
ID 16 <LutherBrid.SG.ir.O> - ID 289 <ArbroathTi.SG.ir.P> <4 times>
ID 16 <LutherBrid.SG.ir.O> - ID 328 <Park.Q.15> <11 times>
ID 29 <Apigill.SG.ir.O> - ID 38 <Corpach.SG.ir.O> <1 times>
ID 29 <Apigill.SG.ir.O> - ID 163 <RenfrewTid.SG.ir.P> <3 times>
ID 29 <Apigill.SG.ir.O> - ID 217 <Clachnahar.SG.ir.P> <5 times>
ID 29 <Apigill.SG.ir.O> - ID 289 <ArbroathTi.SG.ir.P> <4 times>
ID 29 <Apigill.SG.ir.O> - ID 328 <Park.Q.15> <11 times>
ID 38 <Corpach.SG.ir.O> - ID 163 <RenfrewTid.SG.ir.P> <4 times>
ID 38 <Corpach.SG.ir.O> - ID 217 <Clachnahar.SG.ir.P> <5 times>
ID 38 <Corpach.SG.ir.O> - ID 289 <ArbroathTi.SG.ir.P> <5 times>
ID 38 <Corpach.SG.ir.O> - ID 328 <Park.Q.15> <15 times>
ID 100 <Newcastlet.SG.ir.O> - ID 328 <Park.Q.15> <5 times>
ID 163 <RenfrewTid.SG.ir.P> - ID 168 <TarbertTid.SG.ir.P> <4 times>
ID 163 <RenfrewTid.SG.ir.P> - ID 217 <Clachnahar.SG.ir.P> <12 times>
ID 163 <RenfrewTid.SG.ir.P> - ID 289 <ArbroathTi.SG.ir.P> <6 times>
ID 163 <RenfrewTid.SG.ir.P> - ID 328 <Park.Q.15> <30 times>
ID 168 <TarbertTid.SG.ir.P> - ID 217 <Clachnahar.SG.ir.P> <4 times>
ID 168 <TarbertTid.SG.ir.P> - ID 328 <Park.Q.15> <12 times>
ID 217 <Clachnahar.SG.ir.P> - ID 289 <ArbroathTi.SG.ir.P> <9 times>
ID 217 <Clachnahar.SG.ir.P> - ID 328 <Park.Q.15> <39 times>
ID 289 <ArbroathTi.SG.ir.P> - ID 328 <Park.Q.15> <24 times>

Multiple Cooccurrence:
ID 16 <LutherBrid.SG.ir.O> ID 29 <Apigill.SG.ir.O> ID 38 <Corpach.SG.ir.O> ID 163
3 <RenfrewTid.SG.ir.P> ID 168 <TarbertTid.SG.ir.P> ID 217 <Clachnahar.SG.ir.P> I
D 289 <ArbroathTi.SG.ir.P> ID 328 <Park.Q.15> ID 100 <Newcastlet.SG.ir.O>
```

Some of additional information is printed out on the screen during run-time, which is not as important as the final output. If you would like to see some of that information, refer to the Maintenance manual to adjust the output display.

Appendix B: Maintenance Manual

The source code of the DDCA versions (together with the data files) could be found in the 'source' folder in project submission.

Instructions on how to compile and run the algorithm

There are two different variations of the DDCA algorithm: ddcaPORT and ddcaSEPA.

Use the ddcaPORT for the analysis of port scanning data.

Use the ddcaSEPA for the analysis of SEPA river data.

The two source codes are compiled and run in the exact same way.

To compile and run the code on a Windows environment it is possible to use any of the available IDEs for C/C++ that support GCC or MinGW compilers.

Some options:

Bloodshed Software – DEV-C++

URL: <http://www.bloodshed.net/devcpp.html>

Code::Blocks

URL: <http://www.codeblocks.org/>

Make sure that the C source file is located in the same folder with the data files that you want to analyse. As the programs consist of a single C file, most IDEs will have a similar way of viewing, building and running the program.

- 1) Open the IDE and in the file menu click "Open file";
- 2) Enter the folder ('ddcaPORT' or 'ddcaSEPA') containing the C source code and select this file;
- 3) The code will be displayed in a separate window, where you can modify it;
- 4) In the options select "Compile and Run" or "Build and Run";
- 5) The program will start in a command/terminal window.

Some IDEs (like code::blocks) require creating a project first:

- 1) Create a new C Project;
- 2) Add the C source code to the project;
- 3) Copy the data files (logs that you want to analyse) in the working directory;
- 4) “Build & Run” the project;
- 5) The program will start in a separate command/terminal window.

To compile and run the source code on a Unix/Linux environment:

Make sure that the C source file is located in the same folder with the data files that you want to analyse.

- 1) Open the terminal in the folder ('ddcaPORT' or 'ddcaSEPA') containing the C source code
- 2) Type (depending on the needed source code) the following and press enter:

```
% gcc -o ddcaPORT ddcaPORT.c
```

or

```
% gcc -o ddcaSEPA ddcaSEPA.c
```

- 3) Then type (depending on the needed source code) the following and press enter:

```
% ./ddcaPORT
```

or

```
% ./ddcaSEPA
```

- 4) The program will start in the same terminal window

The User Manual provides a guide for using the program.

Organisation of System Files, including directory structures

pilipenko_denis/

executable/

ddcaPORT/ - folder containing executable files for port scanning DDCA

ddcaPORT.exe

ddcaPORTunix

data files ('s1_norm.log', 's2_norm.log', 's3_norm.log', 's4_norm.log')

ddcaSEPA/ - folder containing executable files for SEPA river data DDCA

ddcaSEPA.exe

ddcaSEPAunix

data files ('sepa1.log', 'sepa2.log', 'sepa3.log')

source/

ddcaPORT/ - folder containing source code files for port scanning DDCA

ddcaPORT.c

data files ('s1_norm.log', 's2_norm.log', 's3_norm.log', 's4_norm.log')

ddcaSEPA/ - folder containing source code files for SEPA river data DDCA

ddcaSEPA.c

data files ('sepa1.log', 'sepa2.log', 'sepa3.log')

ddcaORIGINAL/

ddca.c – original DDCA algorithm used for modification

README.TXT

MAINTENANCE MANUAL

Space and Memory Requirements

2 GB of RAM, 100 MB of space would be recommended.

List of source code files, with a summary of their role

File	Description
ddca.c	Original DDCA implementation by J. Greensmith (for comparison)
ddcaPORT.c	Modified DDCA for port scanning data analysis
ddcaSEPA.c	Modified DDCA for SEPA river data analysis

Program Flow

Both of the modified algorithms (ddcaPORT and ddcaSEPA) have a similar program flow in terms of functions and methods. This section provides a general description of the program flow from a technical side.

Initialisation

Most of the algorithm's functions are called from the main() method. After setting up the number of DCs in the populations, the migration threshold, the number and names of the streams (data logs), the algorithms initialise the cells in the populations by calling the initDC() function for every cell. The detection phase starts once all the cells in all the populations have been initialised.

Detection

Input_line() and easy_explode() functions are responsible for signal input parsing and processing. In case of a 'signal' the do_signals() function is called for calculating the K and CSM values and updating all DCs in the populations by calling the update_DC() function. In case of an antigen the do_antigen() function is called that records the given antigen's occurrence for a specific DC. These functions are also responsible for collecting timestamps. After a DC's lifespan reaches zero, the log_antigen() function is called that updates the global antigen profile based on the cell's K value and collected antigens.

Analysis

After all the data files have been processed, every DC in the populations are processed by the same log_antigen() function. The DC statistics are printed afterwards by the dc_stats() function. The result() function calculates the MCAV and K values of every antigen and

updates the 'dangerous' antigen profile (agsD and agsC for SEPA data). Finally, the Dual and Multiple Co-occurrence analyses are performed by the `cooccurrence()` and `multiCooccurrence()` functions respectively. In the end, the results are printed out on the screen and saved in a file "output.txt" the `printOutput()` function.

Changing the Internal Parameters

Changing Time Windows' values

These values are defined in the beginning of the algorithm:

TIME_WIN_SAVE – time window for saving timestamps

TIME_WIN_CO – time window for checking the co-occurrence of the dangerous antigens

Changing the Thresholds for the River Height and Rapid Height Changes

These thresholds specify if the current river height or river height changes should be considered dangerous or normal. The values can be changed in the `do_signals()` function.

Observing the Output during Run-Time

Different types of results are printed out during run-time, however, because some of this output contains too much information (which is also not as significant as the analysis results), most of that data is overrun by the other printouts. It is possible to stop the process at any point if you are particularly interested in a specific piece of information. One approach is to put a `getchar()` function at the relevant spot, which waits for the user input to continue the process.

Suggested positions:

- After the `result()` call in the `main()` function to display the K and MCAV values of every antigen in the profile (not just anomalous);
- After the `dc_stats()` calls in the `main()` function to display the DC statistics;

Adding an extra data stream

To add an extra data stream, several extra values need to be initialised first (done in the beginning of the algorithm):

- A DC structure *cell;
- A cell_index for that stream.

Afterwards, the main() method has to be modified to account for an extra data stream processing:

- Put the correct number of streams for verification of the user input;
- Initialise an extra char* filename and an additional FILE;
- Initialise the extra cell_index and DC population structure cell;
- Add an enquiry for the name of the extra stream (same as for the other streams);
- Add an additional iteration (reading buffer) process for the extra stream;
- Add an extra log_antigen() and dc_stats() processing for the extra stream.

These instructions could be easily completed by examining how processing of the initial four streams is implemented. It is important to make sure that all six points have been completed in order for the algorithm to work properly.