Decentralised Detection of Emergence in Complex Adaptive Systems

Eamonn O'Toole

A Dissertation submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

November 2015

Declaration

Eamonn O'Toole

Dated: November 2, 2015

Acknowledgements

The African proverb states that “it takes a village to raise a child” and the same sentiment certainly applies to my career as a researcher. First, and foremost, I want to thank my supervisor, Prof. Siobhán Clarke for offering me support, expertise, encouragement and good humour during the process of my Ph.D. studies. I also owe an huge debt of gratitude to Vivek Nallur, for his enthusiasm, guidance and willingness to discuss any topic at the drop of a hat. A big thank you to Julien Monteil for his advice on matters statistical and signal-filtering, as well as his thoughts on the dynamics of traffic-flow. Thank you to everybody in DSG, especially those of you who shared the office with me over the years. A very special thank you to my colleagues Saeed Hajebi and Amit Raj, for joining me on the both the Ph.D. and TRANSFoRm adventures, providing somebody to talk to about both. I’d also like to thank my friends away from Trinity, especially Simon Tuohy, Will O’Keeffe, Niall Curtin and Declan Bourke, for encouraging me and reminding me that there is more to life than academia.

Thank you to my entire family, especially to my parents who have supported and encouraged my education from the very beginning. I dedicate this thesis to them.

Finally, Sinead, you have been my rock throughout the past 5 years. Thank you for your patience, love and constant support. This would not have been possible without you.

Eamonn O’Toole
University of Dublin, Trinity College
November 2015

Abstract

Emergence is a hallmark of Complex Adaptive Systems (CAS), where non-deterministic interactions between agents can give rise to emergent behaviour or properties at the system level. The nature, timing and consequence of emergent behaviour are fundamentally unpredictable and may be harmful to the system or individual agents. This unpredictability, coupled with the decentralised structure of these systems, means that detecting emergence at run time presents a significant challenge. No single component of a decentralised system can possess a global view of system state, or reliably know in advance the relevant system properties that indicate emergence.

Existing approaches to emergence detection have used statistical analysis of system variables that represent global features of the system. These techniques depend on a centralised system monitor with access to information on global system state. However, neither of these assumptions are met in CAS, as these are distributed and composed entirely of decentralized autonomous agents. Additionally, these approaches require prior knowledge of the system properties relevant for emergence detection, which may not be obvious or available for all systems. Other approaches use formal methods to define and predict emergence, but are intended for use at design time or are limited to well-defined, closed systems.

This thesis presents Decentralised Emergence Detection (DETect), a novel distributed algorithm that enables constituent agents to collaboratively detect emergent events in CAS. The main contribution of DETect is that it does not require any centralised controller or system monitor, and instead runs locally on each agent. In addition, DETect uses only the available information from an agent's local environment at run time to facilitate detection. DETect relies on the feedback that occurs from the system level (macro) to the agent level (micro) when emergence is present. This feedback constrains agents at the micro-level, and results in changes occurring in the relationship between an agent and its environment. DETect uses statistical methods to automatically select the properties of the agent and environment to monitor, and tracks the relationship between these properties over time for changes. When a significant change is detected, the algorithm uses distributed consensus to determine if a sufficient number of agents have simultaneously experienced a similar change, before a shared conclusion is reached that emergence has occurred. On agreement of emergence, DETect raises an event, which its agent or other interested observers can use to act appropriately.

The evaluation of DETect uses three multi-agent simulation case studies: flocking, pedestrian counter-flow and traffic. Each simulation model exhibits emergence allowing performance to be assessed across a range of system scales. The case studies are structured to verify the efficacy of DETect in three competences: autonomously selecting what properties to monitor, detecting feedback from emergence and forming consensus among agents on the presence of emergence. Performance is evaluated for detection of both the formation and evaporation of emergence in each system. The results of these studies demonstrate that DETect generally achieves its design objectives, facilitating the decentralised detection of emergence formation in each case study. However, the general applicability of DETect during periods of emergence evaporation is limited, with successful detection achieved in the pedestrian and traffic case studies only.

Publications Related to this Ph.D.

1. O'Toole, Eamonn, and Siobhán Clarke. "Dynamic forecasting and adaptation for demand optimization in the Smart Grid" *Proceedings of the First International Workshop on Software Engineering Challenges for the Smart Grid.* IEEE Press, 2012.

2. O'Toole, Eamonn, Vivek Nallur, and Siobhán Clarke. "Towards Decentralised Detection of Emergence in Complex Adaptive Systems" *Self-Adaptive and Self-Organizing Systems (SASO), 2014 IEEE Eighth International Conference on.* IEEE, 2014.

3. O'Toole, Eamonn, Vivek Nallur, and Siobhán Clarke. "Decentralised Detection of Emergence in Complex Adaptive Systems" **submitted to** *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* (submitted September 2015).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis presents Decentralised Emergence Detection (DETect), a novel distributed algorithm for decentralised emergence detection at run time in Complex Adaptive Systems (CAS). DETect enables the constituent agents of the system to collaboratively act as detectors of emergence using only locally available information. This is achieved using a combination of statistical and machine learning techniques along with distributed consensus, removing the need for a centralised controller or system monitor. This chapter motivates the work and introduces the principal ideas underpinning emergence. Next, the challenges presented by detecting emergence in CAS are outlined before the goals and contributions of this work are described. Finally, the chapter closes by describing the structure of the rest of this thesis.

1.1 Motivation

Ever increasing power, heterogeneity and interoperability of computational devices means that future large-scale systems will both host, and depend on, an array of Complex Adaptive Systems (CAS). CAS are systems composed of independent heterogeneous agents that interact, adapt and learn [Holland, 1992]. Individual agents are software components, systems or people, each with unique, possibly conflicting, goals. CAS are capable of spanning organisations and large geographic areas with no centralised control or monitoring [Northrop et al., 2006]. They form organically through the non-deterministic interactions of their agents, which means predicting the characteristics and behaviour of these systems with any great degree of accuracy is impossible in advance [Mogul, 2006, Johnson, 2006].

Emergence is a hallmark of such systems [Innes and Booher, 1999, Ottino, 2004], and refers to the appearance of persistent properties and behaviours (emergents) at the system (macro) level. These emergents are caused dynamically from the non-deterministic interactions between entities at the agent (micro) level [De Wolf and Holvoet, 2005]. However, despite being caused by agent interaction, the emergents are novel with respect to the underlying agents, meaning they cannot be reduced to the properties or behaviour of the individual entities [De Wolf and Holvoet, 2005, Gleizes et al., 2008]. Typical examples of emergence include traffic jams, swarm formation of birds and the Internet [Fromm, 2005].

The interactions of agents at the micro-level are said to have causal power on the emergents at the macro-level. However, the macro-level can also impact on the micro-level through downward causation, where the emergent phenomenon constrains the agents at the micro-level [Bedau, 2002]. A car in a traffic-jam provides an illustrative example of this bi-directional relationship between the two system levels. The car contributes to the volume of traffic causing the emergent behaviour, and this emergent behaviour impacts the car by limiting its speed and also, potentially, the route it will take. Therefore, when emergence occurs it is beneficial to be aware of its presence so that its effects can be mitigated if they are harmful (such as a traffic jam) or leveraged if they are beneficial (such as increased security when flocking).

However, the scale, unpredictability and decentralised architecture of CAS means that detecting emergence in these systems presents significant challenges. These challenges are compounded by the nature of emergence, which is itself both dynamic and unpredictable. Existing research cannot meet these challenges as they rely on centralised architectures with a global system view and design time knowledge of the emergent properties to facilitate detection. Instead, effective detection should occur at runtime, be decentralised across the constituent agents and require no prior knowledge of the specific emergent properties and behaviours that may arise in the system.

1.2 Emergence in CAS

The concept of emergence first appeared in philosophy in the time of Socrates and has been studied in a variety of fields such as physics, computer science and social science, ever since [Di Marzo Serugendo et al., 2006]. It is a fundamental concept in the field of complex systems and can be considered a necessary part of what qualifies complex systems as complex [Deguet

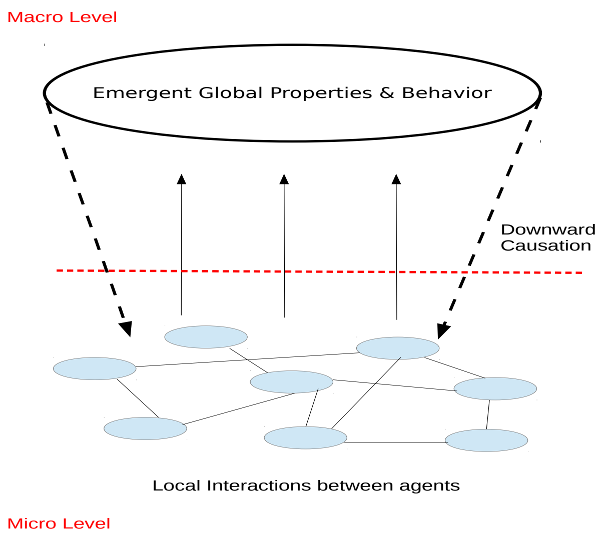
Macro Level
Emergent Global Properties & Behavior
Downward Causation
Local Interactions between agents
Micro Level

Fig. 1.1: **Emergence and the system levels:** *A bi-directional link exists between system levels when emergence is present. The macro-level arises from agent interaction at the micro-level and the macro-level constrains the micro-level through downward causation.*

et al., 2006, Flake, 1998]. Underpinning the concept of emergence is the notion that the whole is greater than the sum of the parts, or to put it another way, that order can arise from chaos [Kubík, 2003]. In CAS, emergence refers to macro-level patterns, structures and properties arising in systems of decentralised interacting agents, where these patterns, structures and properties are not contained in the properties of its parts [Dessalles and Phan, 2001].

This definition highlights a number of characteristics of emergence that should be noted. The first is that these systems are decentralised with no centralised controller or monitor. Second, it is necessary to talk about these systems at two levels when discussing emergence; the level of the individual agent (micro) and the level of the global system (macro). Third, emergence is only possible in systems composed of autonomous agents who interact non-linearly and it is these interactions that give rise to the emergent behaviour or property. Finally, the emergent properties at the macro-level cannot be simplified to a composite of the properties of the micro-level.

The relationship between the two levels is one of interdependence with the micro-level causing the macro, while the macro-level constrains the micro. This two-way relationship between the micro and macro levels is illustrated in Figure 1.1. An everyday example of this is the coherent flocking patterns that flocks of individual birds exhibit, which arise from simple interactions between the individual birds. Once formed, the global flocking behaviour constrains the behaviour of the individual birds that constitute it.

Emergent behaviour and properties can be unexpected, undesirable and difficult to control, for example traffic congestion [Singh et al., 2013]. However, it is also possible that emergence can be beneficial, such as the increased security from predators offered by flocking. In either case, it is desirable to be able to detect emergence when it occurs so that harmful effects can be mitigated or beneficial effects can be exploited. It is common to think of the observer of emergence as being external to the system, looking at the overall system state from a global viewpoint, for example, a human being observing a flock of birds flying in formation. However, cognitive agents, agents with knowledge of their environment and the ability to use past experiences [Olaru and Florea, 2009], can theoretically be used to detect these phenomena if they are themselves part of the system [Dessalles and Phan, 2001]. This is made possible if the agents know what to look for and have a sufficiently broad view of the system.

1.3 Challenges

It is possible for both internal and external system observers to recognise emergence when it occurs and to therefore act as detectors. However, the characteristics of both CAS and emergence means that this task is non-trivial to accomplish. In particular, any emergence detection mechanism faces the following challenges:

- **Challenge 1: Decentralised control and monitoring**

  CAS are decentralised systems capable of spanning organisations and large geographic areas [Holland, 1992, Northrop et al., 2006]. As a result, no single agent of the system is responsible for undertaking or coordinating emergence detection. Additionally, emergence occurs at the macro-level of the system. However, no single agent has immediate access to a global view of the system state and obtaining this information in systems of scale becomes impractical as the system state is constantly changing. Instead, agents may have access to locally available information which is more timely, but is limited in scope and insufficient to form conclusions about emergent behaviour and properties. This requires that detection should be decentralised across all agents with collaboration used to overcome the limited system view of individual agents.

- **Challenge 2: Unpredictable non-linear interactions**

  CAS are composed of many agents that interact non-linearly in ways that are difficult to recognise, manage and predict [Maxwell et al., 2002]. This unpredictability results in unforeseeable connections and subsystems forming organically at runtime. However, the autonomy of the constituent agents means that these systems are dynamic and any connection or subsystem formed remains transient. Agents may move to another part of the system or decide to interact with a different set of agents depending on their individual goals. This presents significant challenges to any collaboration attempt between agents as the set of agents that comprise an individual agent's neighbourhood is constantly in flux.

- **Challenge 3: Knowing what to look for**

  The characteristics of emergence are unpredictable [Stephan, 1999]. This does not mean that these characteristics are not knowable for all systems as, for example, in well understood complex systems such as traffic or flocking, the type of emergence that occurs is understood. However, outside of such systems, the unpredictability of emergence means that it is not possible to know what system configuration constitute emergence. Therefore, knowing in advance what properties to monitor and what indicates the presence of emergence is not possible. As a result, a general method is required to select the important properties to monitor at runtime with minimal design time input.

- **Challenge 4: Transient nature of emergence**

  In systems with emergence, emergents arise and disappear as the system evolves over time [De Wolf and Holvoet, 2005]. An illustrative example is provided by flocking in birds with the emergent flocking behaviour being capable of both forming and evaporating and thus being transient in nature. For appropriate steps to be taken to either mitigate or leverage the new macro-state of the system, an effective detection method should be timely. This means that detection events should occur when the emergence is forming, before the new emergent state of the system is established. Similarly, it requires that evaporation of the emergence should also be detected, especially in situations where the emergent state was beneficial.

1.3.1 Existing Solutions

Existing methods for detecting emergence have used a variety of statistical methods on system variables that represent global features of the system [Bar-Yam, 2004, Mnif and Muller-Schloer, 2006, Fisch et al., 2010, Seth, 2008, Chan, 2011, De Wolf et al., 2005, Grossman et al., 2009, Niazi and Hussain, 2011]. Although these approaches allow detection to occur at runtime, they depend on a centralised architecture, with an omniscient system monitor or controller who has immediate access to the global system state. However, such an approach is not possible in CAS that are distributed, and composed entirely of decentralized autonomous agents. Additionally, these variable based approaches require prior knowledge of the system properties that describe emergence and therefore should be monitored. This assumption limits their applicability to systems with well defined and understood emergent properties. However, these properties may not be obvious or available for all systems, especially in CAS which are dynamic and unpredictable.

Other approaches use formal language modelling and simulation of large multi-agent systems to either detect emergence [Randles et al., 2007, Ciancia et al., 2014, De Angelis and Di Marzo Serugendo, 2015], or *predict* whether emergence *can* occur in a system [Moshirpour et al., 2012, Kubík, 2003, Teo et al., 2013]. These methods require both the system and what constitutes expected emergent behaviour to be defined and understood at design time. This requirement makes them unsuitable for CAS, which form organically with unpredictable composition and behaviour and thus, may not be practicably or effectively modelled or simulated in advance in all cases. Moreover, these approaches are centralised or require some agent, either external or part of the system, to acquire a global view of system state. Finally, the transient nature of emergence, forming and evaporating as the system evolves, is not addressed by these approaches. Therefore, detecting unforeseen or unintentional emergence when it occurs at runtime in decentralised CAS remains an open research problem.

1.4 Thesis Approach

A novel distributed algorithm, DETect, is designed to enable constituent agents to collaboratively detect emergent events in CAS, without the need for a centralised controller or system monitor.

Assumptions For the agents and environment in which emergence detection is occurring, this thesis makes the following assumptions:

- System time on all agents in the system is synchronized and all agents are assumed to make decisions and record variable observations simultaneously at fixed time intervals.
- Agents are assumed to be to be failure free. Therefore, issues arising from agents not being able to contact their one-hop neighbours, or agents receiving incomplete or inaccurate information is not addressed. In simulations in this thesis, agents are mobile with their one-hop neighbours constantly changing based on other agents within a fixed radius.
- Agents have access to a set of variables that describe their own state and a set of variables that describes the subset of the environment that they care about. The variables that compose these sets is assumed to remain static once the agent is initialised. Additionally, agents can provide both sets of variables to DETect and inform DETect what set describes the agent's state and what set describes the environment's state.
- The agents or the system contains an interested party, such as an adaptation manager, to receive detection events from DETect and respond appropriately. DETect fits into the monitoring stage of a MAPE-K loop [Kephart and Chess, 2003], so planning and executing comprehensive behavioural adaptations in response to emergence is beyond the scope of this research.

Hypothesis This thesis investigates how to facilitate the detection of emergence at runtime in CAS without using a centralised system monitor. The hypothesis of this research is as follows: *When emergence is forming or evaporating in a system, a significant proportion of the constituent agents will simultaneously experience a change in the statistical relationship between themselves and their environment. By sharing this experience, agents can collaboratively act as detectors of the emergent event.*

Basic Idea The presence of emergence in a system results in feedback from the macro-layer to the micro-layer, through downward causation, that constrains the agents in the system [Chalmers, 2006, Bedau, 2002]. As a result, the relationship individual agents have with their local environment, including other agents, will be different when emergence is present in the system compared to when there is no emergence. As emergence is global in nature, all agents involved should simultaneously experience a changed relationship as the emergent behaviour forms and evaporates in the system. This hypothesis inspires the conceptual architecture of DETect, illustrated in Figure 1.2, with a modular design used to achieve three key competences.

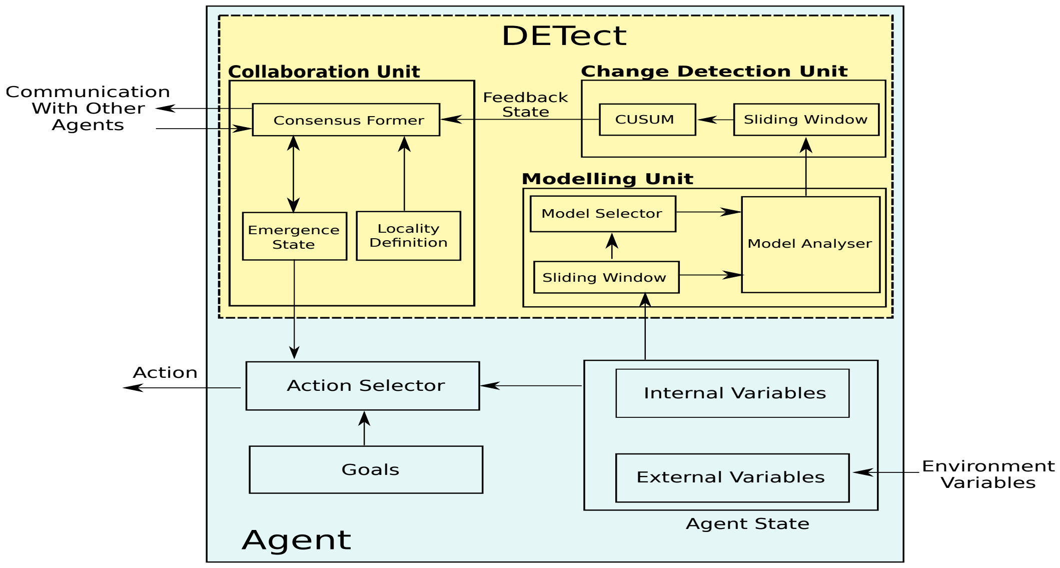

Fig. 1.2: **DETect - Conceptual Architecture:** *DETect is composed of three units, each of which delivers one of three key competences; modelling, change detection and collaboration.*

The *Modelling Unit* provides agents with a general method of modelling their relationship with their local environment. It achieves this using properties of the agent (internal variables) and properties of the agent's environment (external variables), with a subset of these variables autonomously selected at runtime to compose the model. The statistical properties of the model are periodically measured as the system executes, with recent observations used to establish an expected baseline for the agent's current relationship with its environment. The *Change Detection Unit* provides agents with an on-line means of detecting a sudden and statistically significant change in the relationship, that may indicate that the agent is experiencing feedback from emergence. However, it is not possible for any single agent to independently conclude the presence of emergence due to the limited scope of their system view. Therefore, the *Collaboration Unit* uses a distributed consensus algorithm to determine the proportion of agents simultaneously experiencing a similar change with the existence of an emergent event concluded if the proportion is sufficiently high. Once this agreement is reached, an event is raised which can be used as input by an adaptation manager to react appropriately.

1.5 Thesis Contribution

This thesis investigates how the constituent agents of CAS can be facilitated to act as detectors of emergent events when they occur at runtime. This research contributes to the body of knowledge by providing:

Decentralised detection across agents The primary contribution of this thesis is to facilitate decentralised detection of emergence. Existing detection approaches rely on centralised architectures and knowledge of the global system state. However, neither of these are possible in CAS and therefore detection should be distributed across the constituent agents. This is achieved by enabling individual agents to detect the feedback from emergence when it occurs through alterations in their relationship with their local environment. Distributed consensus is then used to share information across agents to allow agents to collaboratively build wider views of the system state. This ensures that conclusions on emergence are not based on the limited local view of a single agent.

Automated model selection Existing approaches to emergence detection require significant design time knowledge of the properties of the system that describe emergence and should be monitored. This limits their utility to only those systems they are designed for. This thesis describes a general statistical method that autonomously selects useful variables from the agent and its environment to monitor as the system evolves. These variables are used to model the relationship between the agent and its environment with this model used to facilitate detection of feedback from emergence to the agent. As a result, the requirement for specific design time knowledge of the emergent properties and what should be monitored is removed.

Detection of both formation and evaporation of emergence Emergence is transient in nature, both forming and evaporating as the system evolves over time. Existing detection approaches do not address this transience or consider the timeliness of detection, instead focussing on detection only when the emergent behaviour is established. This limits the utility of detection as emergence may have already evaporated by the time any adaptation action is taken. This thesis investigates how to detect both formation and evaporation periods of emergence to enable appropriate actions to be taken before the new macro-state becomes established. The proposed approach uses a sliding window model of the recent history of the agent's relationship with its environment. Each new evaluation of this relationship is compared against this window to determine if the statistical properties have significantly changed. This new evaluation is added to the model with the oldest entry removed, ensuring that future comparisons will be based on an up-to-date baseline. This allows the baseline to be updated gradually as the system evolves, with only sudden significant changes triggering suspicion. As a result, detection events are raised during both emergence formation and evaporation periods when the state of the system is in flux.

1.6 Thesis Structure

The remainder of this thesis is structured as follows,

Background and Related Work: Chapter 2 outlines the features and different types of emergence and describes how these allow emergence to be observed both outside and inside the system. Next existing detection approaches are presented and categorised under three broad types i) variable based ii) formal language/model based and iii) event based. The analysis of these approaches is rooted in the challenges presented by emergence detection in CAS, with particular emphasis placed on when the detection occurs, what information is required and the extent to which they are decentralised.

Design: Chapter 3 returns to the challenges presented by emergence detection in CAS, outlined in Chapter 1, and describes the design objectives, system model, and design decisions of this thesis. Next, the chapter describes in detail the proposed distributed algorithm for emergence detection which provides three core competences; model selection, feedback detection and consensus formation among agents on the existence of an emergent event in the system. The chapter concludes by explaining how this solution addresses the design decisions.

Implementation and Simulation Environment: Chapter 4 describes the implementation of the DETect algorithm using NetLogo, a multi-agent modelling platform, Java and R. Next, it outlines how DETect is integrated into agents in three multi-agent models to provide a simulation platform on which evaluation can be performed.

Evaluation Chapter 5 evaluates the performance of DETect in each of the three core competences. It first describes the experimental set-up, with a case study of three multi-agent systems that exhibit emergence used to evaluate performance across systems of different scales. The second part of the chapter presents and analyses the results of this study.

Conclusion: Chapter 6 summaries the thesis by discussing the strengths and weaknesses of the DETect algorithm and identifying a number of potential areas for future work.

1.7 Chapter Summary

Emergence is a hallmark of CAS and refers to the appearance of persistent properties and behaviours at the macro-level of the system. It arises through the non-deterministic interactions of autonomous decentralised agents and its characteristics are unpredictable in advance. Emergent behaviour and properties can be both harmful and beneficial to the constituent agents and the system as a whole. Therefore, emergent behaviour and properties of a system should be detected when they occur at runtime so that appropriate steps can be taken to mitigate harmful effects or leverage beneficial emergent behaviour.

CAS lack any centralised controller or monitor with no individual agent possessing a complete view of the global system state. These characteristics present a significant challenge to detection mechanisms as emergence is unpredictable and occurs above the level of individual agents. Existing detection approaches are ill-equipped to meet these challenges as they rely on centralised architectures, advanced knowledge of the emergent properties or knowledge of the global system state to facilitate detection.

This thesis presents DETect, a novel distributed algorithm for decentralised emergence detection in CAS. DETect enables the constituent agents of the system to collaboratively act as detectors using only locally available information. DETect relies on downward causation from the macro-level to the micro-level when emergence is present in the system, that naturally constrains the agents. DETect builds on the hypothesis that this feedback can be detected by changes in the relationship between an agent and its environment, characterised by variables that describe the agent and variables that describe its local environment. The following chapters describe how DETect achieves this using a combination of statistical and machine learning techniques to autonomously model this relationship and monitor it over time for changes. Finally, distributed consensus is used to share these changes with other agents and therefore build a shared consensus on the presence of an emergent event.

Chapter 2

Background and Related Work

This chapter presents background and related work. It begins by outlining the history of the concept of emergence and its accepted characteristics. This is followed by a description of the different types of emergence that exist and their consequence for both the system and would-be observers of the emergent phenomena. The next section describes the systems that are the focus of this thesis, Complex Adaptive Systems, and outlines the various techniques used to investigate them. Following this necessary background, the last part of the chapter discusses the state of the art in emergence detection techniques and the gaps left by existing solutions.

2.1 Emergence

This section introduces the concept of emergence, describing its characteristics and the tools available to study it. In addition, the different types of emergence that can occur are outlined, illustrating how emergence can be observed and enabling the research described in this thesis to be situated within a larger context.

2.1.1 Characteristics of Emergence

Emergence has a long history in the philosophy of science, appearing in a variety of fields such as physics [Gemmer et al., 2009], biology [Cagan and Ready, 1989] and complexity theory [Sawyer, 2005]. This diversity of study is perhaps reflected in the fact that there exists no universally accepted definition of the term [Olaru and Florea, 2009], with both its characteristics and its very existence generating much debate. The concept underpinning emergence dates back to ancient Greek philosophy, when Aristotle claimed that, for substances, "the whole is something over and above its parts, and not just the sum of them all" [Ross, 1997, 1045H8-10]

The term *emergence*, as it is understood today, first appeared in the 19$^{\text{th}}$ century when it was coined by the British philosopher G.H. Lewes to distinguish between chemical compounds created by a chemical reaction [Lewes, 1875]. Lewes referred to compounds that were created in a process that was understood and could be traced back to the underlying components/agents as "resultants". Chemical compounds that could not be reduced in the same way to the sum (or difference) of the agents were called "emergent" compounds, with Lewes stating that they arise "out of the combined agencies but in a way that does not display the agents in action."

Emergence is a complicated concept and does not fit neatly into any concise definition [Holland, 2000]. Abbot associates it with the appearance of an "epiphenomenon", a phenomenon that can be described independently of the underlying phenomenon that cause it [Abbott, 2006]. De Wolf and Holvoet provide a working definition of emergence based on its use throughout relevant literature [De Wolf and Holvoet, 2005]. The authors state that a system exhibits emergence "when there are coherent emergents at the macro-level that dynamically arise from the interactions between the parts at the micro-level. Such emergents are novel w.r.t. the individual parts of the system". Emergents refer to behaviours, patterns, structures and properties.

Greater clarity is provided by examining the characteristics of emergence and the conditions necessary for it to exist. Goldstein states that emergence can only occur in systems that are non-linear, self-organising and do not exhibit equilibrium or homeostasis [Goldstein, 1999]. De Wolf and Holvoet augment their working definition by outlining 8 characteristics that identify behaviour and properties as emergent [De Wolf and Holvoet, 2005]. These are:

- **Micro-Macro Effect:** The most fundamental characteristic is that, when emergence arises, it becomes necessary to talk about the system at two levels, the macro and the micro. The macro-level refers to the global system level where the emergent properties and behaviour are exhibited and can be observed. The micro-level contains the individual agents or components of the system and it is the interactions of the agents at this level that cause emergence. **Example:** A traffic-jam, which constitutes emergent behaviour at the macro-level, in comparison to the individual vehicles that compose it at the micro-level.

- **Bi-directional Link:** A bi-directional link exists between both levels, with the micro-level said to cause the macro-level and the macro-level constraining the agents at the micro-level through downward causation [Ladyman et al., 2013]. This relationship gives rise to a number of different types of emergence, which are discussed in detail in Section 2.1.3. **Example:** A traffic jam is caused by the set of individual vehicles that compose it. Once it has formed, the congestion influences the speed and direction of the individual vehicles.

- **Interacting Parts:** The individual agents at the micro-level must interact with one another for emergence to be possible. It is not sufficient for the system to be composed by autonomous agents that act in isolation of one another. Furthermore, it is necessary for the interactions and behaviour of these agents to be non-linear for emergence to arise [Di Marzo Serugendo et al., 2006]. **Example:** The traffic jam is caused by individual vehicles interacting with one another, other road users and traffic signals.

- **Dynamical:** Emergents are not perpetual in the context of the system and instead arise as the system evolves in time [Holland, 2000]. Emergence is a new property or behaviour that becomes possible at a certain point in time. This new behaviour is said to persist as a coherent identity over time. However, when the circumstances of its existence are no longer present, the emergence may evaporate and disappear from the system. **Example:** The traffic jam forms and evaporates.

- **Decentralised Control:** Control of the agents must be decentralised in the system with emergence not being possible if agents do not act autonomously [Goldstein, 2000]. This means that the agents, using only local information and mechanism, create and influence the emergence, with no centralised control. **Example:** In the traffic jam, although they are influenced by one another and by traffic signals, vehicles act autonomously, each with their own goal and behaviour.

- **Coherence:** Emergents appear as coherent wholes, maintaining an identity while they persist in the system. Goldstein notes that this coherence spans and correlates the separate lower-level agents into a higher-level unity [Goldstein, 1999]. **Example:** The traffic jam spans the individual vehicles that compose it and can be observed as a coherent behaviour.

- **Robustness & Flexibility:** As the system is decentralised, there is no single point of failure, so the emergent behaviour or property can survive if individual agents fail. **Ex-**

Table 2.1: Detection challenges and emergence characteristics

Detection Challenge	**Associated Emergence Feature**
Decentralised control and monitoring	- Decentralised Control
Unpredictable non-linear interactions	- Interacting parts
Knowing what to look for	- Radical Novelty - Micro-Macro Effect
Transient nature of emergence	- Coherence - Dynamical

ample: The traffic jam will persist if an individual vehicle leaves the system, for example, by parking.

- **Radical Novelty:** The emergent behaviour is novel with respect to the individual behaviour at the micro-level and is not reducible to the agents whose interactions create it. As a result, emergence is also unpredictable, with Goldstein noting that emergent phenomena are neither "predictable from, deducible from, nor reducible to" the constituent agents of the system [Goldstein, 1999]. This unpredictability is only absolute the first time an emergent phenomenon is observed, however the non-linear behaviour of these systems means that there will be some differences in the emergents each time they arise. **Example:** The concept of a traffic jam does not apply at the level of individual agents.

Emergent behaviour can be both beneficial or harmful to the system and the constituent parts [Paunovski et al., 2008]. For example, flocking behaviour provides greater protection from predators, while traffic-jams cause pollution and cost time. Mogul points out that even when emergence is not inherently bad, it is unpredictable and in a computer system's context, unpredictability is bad [Mogul, 2006]. However, the characteristics of emergence outlined above means that detecting emergence is a difficult task. This is illustrated in Table 2.1, where each emergence detection challenge outlined in Section 1.3 is mapped to the underlying characteristic of emergence that motivates it.

2.1.2 Studying Emergence

The idea of radical novelty and irreducibility of the emergent phenomena has generated debate since Lewis first coined the term [Lewes, 1875]. This resulted in the development of two schools of thought on emergence, proto-emergentism and neo-emergentism [Goldstein, 1999]. Proto-emergentism, also known as the philosophical approach, viewed the emergence process as a black box and was initially the most popular outlook. It claimed that only the inputs and outputs at the lowest level can be discerned and the process of how the inputs become the outputs was fundamentally un-knowable. This meant that the emergent process could not be studied scientifically as the effects were almost "miraculous" in nature [Müller-Schloer and Sick, 2008].

Since the 1930s, neo-emergentism, also known as complexity theory, has become the dominant outlook [De Wolf and Holvoet, 2005]. Neo-emergents seek to make emergence less mysterious with the aim of developing tools and methods to understand and reproduce the process which leads to emergence in systems. As stated by Miller and Page, the field of complex systems must "direct its flight from wonder toward discoveries that make the wonderful and complex understandable" [Miller and Page, 2009]. Complexity theory is composed of a number of different research fields, each approaching the issue from a unique standpoint or concentrating on different types of complex systems. Goldstein describes four broad research fields that comprise complexity theory [Goldstein, 1999]. These are:

- **Far from equilibrium thermodynamics** is the study of thermodynamic systems that are constantly changing with time due to external energy input [Nicolis, 1989]. Arising out of these far-from-equilibrium conditions are dissipative structures that are called emergents. These structures are complex with interacting particles exhibiting long term correlations.
- **Synergetics** is an interdisciplinary field, founded by Hermann Haken [Haken, 1983], which studies the formation of self-organizing patterns and structures in open systems. Order parameters are a core concept in this field. They are a set of collective variables which exist when the system becomes unstable and they influence which coherent macro-level phenomenon the system exhibits.
- **Non-linear dynamical systems theory** is a field of mathematics used to describe the behaviour of non-linear complex systems [Devaney et al., 1989]. Its study of emergence is centred on the concept of attractors, which are specific behaviours that the system evolves to. One example is a strange attractor which was defined by Newmann and claimed to classify a truly emergent phenomenon [Newman, 1996]. It is said to exist when a system's long term behaviour is apparently random yet still exhibits some degree of organisation.
- **Complex Adaptive Systems Theory** was made famous by the Santa Fe Institute in

> the early 1990s [Waldrop, 1993, Gell-Mann, 1994, Holland, 1992]. Anderson notes that complex adaptive systems can be investigated either by studying real examples such as the brain, or by building them in computer systems using artificial entities and artificial intelligence [Anderson, 1999].

The work in this thesis is situated in the field of Complex Adaptive Systems Theory. A detailed discussion of Complex Adaptive Systems (CAS) and how they can be modelled and simulated using multi-agent systems (MAS) is provided in Section 2.2.

2.1.3 Types of Emergence

Downward causation and the identity of the observer of emergence have been used to identify a number of different emergence types. Understanding the distinction between these types is necessary to understand the context in which emergence can be detected and by whom.

2.1.3.1 Downward Causation Based Classification

Downward causation is the "raison d'etre" of the study of emergentism [Kim, 1992]. It refers to the causal power the emergent phenomena exerts on the constituent parts at the micro-level and it is chiefly because of this causal influence that researchers wish to understand the process. Its existence creates a bi-directional link between the two levels of the system, with the micro-level causing the emergents at the macro-level and the emergents subsequently constraining or influencing the constituent parts. Kim criticised the coherence of the notion of downward causation, claiming there was something "circular" about it [Kim, 1999]. Despite this criticism, downward causation is a widely accepted feature of emergence [Heylighen, 2001, El-Hani and Pihlström, 2002, Chalmers, 2006] and it is illustrated, for example, in the relationship between an individual car and a traffic jam.

Bedau highlights that emergent phenomena are generated by and constituted from the underlying processes, while simultaneously being somehow autonomous from those processes [Bedau, 1997]. He argues that these two hallmarks seem to make emergent phenomena inconsistent or illegitimate and that any philosophical defence of emergence requires that this seeming contradiction be addressed. To achieve this, he proposed a distinction between *weak* and *strong* emergence before later adding *nominal* emergence [Bedau, 2002].

Nominal Emergence is the simplest notion and refers to any property that can be possessed by macro phenomena that cannot be possessed by agents at the micro-level. The weakness of this definition is its breadth, essentially representing a super-set of all phenomenon that appear at the macro-level of a system. For instance, it would include both emergents and resultants referred to by Lewes [Lewes, 1875].

Strong Emergence is the most stringent type of emergence identified by Bedau, requiring that the emergents are supervenient and have irreducible causal-powers. To parse this, supervenient refers to the the emergent property being caused by or determined by the micro-level of the system. Causal-powers means that the emergent properties at the macro-level causally affects both the macro-level itself and the micro-level, through downward causation. However, in strong emergence the causal powers cannot be reduced to the causal powers exerted by the micro-level. This idea of emergence existed before Bedau, and is criticised for the sense that the causal powers of the macro-level are somehow gotten for free. As Kauffman [Kauffman, 1995] points out, "what extra can be in the whole that is not in the parts"? Bedau asserts that strong emergence should be embraced if there exists compelling evidence to support it, but concludes that the elusiveness of its causal powers means that it cannot be studied scientifically.

Weak Emergence is an intermediary level between nominal and strong emergence [Bedau, 2002]. A phenomenon is weakly emergent if it can be derived from the dynamics of the micro-level of the system, but only through simulation. Downward causation still exists in weak emergence however, it differs from strong emergence in that weak emergent causal powers can be explained from the causal powers of the micro-level components, but only in a complicated fashion (simulation). This complicated fashion means that the phenomenon cannot be explained merely as a resultant of the micro-level and, therefore, the phenomenon is more than nominally emergent. Bedau identifies ocean waves, traffic jams and vortexes as plausible candidates of weakly emergent phenomena [Bedau, 2002].

2.1.3.2 Observer Based Classification

The observer of the emergent behaviour or property is a key consideration when considering emergence. For example, the novelty of emergent behaviours and properties to the observer has been used by Ronald et. al as an attempt to design a formal test for emergence [Ronald et al., 1999]. De Haan describes three types of emergence that are dependent on the location and role of the observer of the phenomenon [De Haan, 2006].

Discovery Emergence is emergence viewed by an external observer of the system. The property or behaviour must be irreducible to the constituent parts and only attributable to the system as a whole. The observer does not need to view or consider the micro-level components to observe the emergent behaviour or property, as the impact of downward causation is not felt at the micro-level. De Haan suggests fractal patterns in coastlines or river basins as an example of this emergence.

Mechanistic Emergence is similar to discovery emergence in that it once again involves an external observer. However, the key difference is that the observer can no longer just reference the macro-level alone to describe the emergent behaviour and must also make reference to the micro-level. The reason for this is that the components of the system are not oblivious to the emergent phenomenon and instead are influenced by it through downward causation. An example of this type of emergence is flocking behaviour in birds or the exchange value in financial markets.

Reflective Emergence refers to the case where the observers of the emergence property are themselves part of the system, so no external observer is required. The constituent agents are able to observe the emergent behaviour or properties they produce and they use this information to adapt or alter their behaviour as a result. This emergence results in the systems as a whole becoming reflexive, a feature that also makes it the most philosophically troubling. This occurs because the macro-level now has a causal affect on the agents on the micro-level, causing the agents to learn and adapt their behaviour in response to the detected emergent behaviour. This adapted behaviour can in turn impact the emergent behaviour closing the circle of causality between levels.

Muller uses the identity of the observer to distinguish between weak and strong emergence in a manner similar to De Haan's mechanistic and reflective emergence [Muller, 2004]. **Weak emergence** refers to scenarios where the observer is external to the system looking at the overall system state from a global viewpoint, for example, a human being observing a flock of birds flying in formation. In contract, **strong emergence**, occurs when the agents themselves are the observers of emergence, identifying a phenomenon at the macro level that represents an evolution in the system they participate in. Returning to the flocking example, individual birds would identify the emergence of the flocking behaviour. However, the agents' ability to act as emergence detectors requires that their view must be sufficiently broad to be capable of identifying the phenomenon as global rather than local.

2.1.3.3 Hybrid Classification

A hybrid classification is provided by Fromm in the context of multi-agent systems, with both feedback from downward causation and the role of the observer explicitly incorporated into a taxonomy [Fromm, 2005]. This categorisation is a more general evolution of that proposed by Bar-Yam, who outlined a mathematical theory for representing emergent properties of a system by measuring entropy at different levels of abstraction [Bar-Yam, 2004]. Fromm identifies 4 broad categories of emergence, each of which can contain sub-types.

Type-I emergence corresponds to nominal emergence described by Bedau [Bedau, 2002], with no downward causation from the macro-level to the micro-level. It has two sub-types, the first of which refers to scenarios where the emergence is an intended emergent property of the planned interactions of the parts. An example is the function of a machine, which is an emergent property of its components. The second sub-type is when the emergence is unintentional, such as the thermodynamic properties of pressure, volume and temperature.

Type-II emergence involves downward causation from the macro-level of the system. Fromm refers to this as weak emergence and provides how a shoal of fish influences the motions of each participating animal as an example. This type of emergence is broadly predictable however, the causal effect of both system levels on one another means that it cannot be predicted in every detail. The first sub-type is when the feedback from the macro-level is negative and acts to constrain the actions of the agents, such as ant colonies and flocking behaviour. A second sub-type refers to when the feedback is positive, for example a stock market bubble, with agents imitating others instead of acting independently.

Type-III emergence refers to scenarios with multiple feedbacks, learning and adaptation. This type of emergence is chaotic and not predictable due to the complex way these feedback loops interact across levels. The first sub-type refers to a combination of feedbacks similar to those described in Type-II. These can cause chaotic or oscillating behaviour such as, for example, in Conway's Game of Life [Gardner, 1970]. The second sub-type is similar to De Haan's reflective emergence [De Haan, 2006], with the agents sensing and reacting to the emergent phenomena.

Type-IV emergence is strong emergence in the classical sense, as described by Bedau [Bedau, 2002]. Strong emergent properties cannot be reduced, even in principle, to the cumulative effect of the component parts. Fromm cites life as a strong emergent property of genes and proteins and culture as a strong emergent property of memes, language and writing [Fromm, 2005]. However, Fromm states that there is nothing magical about the process, and instead it is the result of very complex phenomena occurring at multiple scales.

2.1.3.4 Emergence and this thesis

This section presented a number of emergence classifications based on the extent of downward causation from the macro-level in the system, and the identity and role of the observer of the emergent phenomenon. Fromm's taxonomy combines both considerations and can therefore be considered the most complete categorisation of emergence presented. However, despite this, there is a large degree of agreement across each classification with, for example, Fromm's Type-II and Type-III emergence being respectively comparable to the mechanistic and reflective emergence described by De Haan. These classifications highlight that the observer of emergence can be either outside the system or inside it. External observers can see the entire system and therefore view the macro-level behaviour, while the presence of downward causation provides constituent agents with a means of detecting the presence of emergence.

The focus of this thesis is investigating ways of detecting emergence at runtime without using an external or global observer. As such, the emergence of interest is that which can be sensed by agents through downward causation, with the expectation that by detecting the presence of emergence, agents can take steps to mitigate or leverage its effects. This work is therefore positioned in the gap that exists between Type-II (mechanistic) and Type-III (reflective) emergence, where feedback exists from the macro-level to the micro-level and the observer. Both De Haan and Fromm state that systems capable of Type-III emergence are true examples of complex adaptive systems [De Haan, 2006, Fromm, 2005].

2.1.4 Emergence and Self-Organisation

Self-organisation is a concept that is often conflated with emergence, however there are key distinctions between them [De Wolf and Holvoet, 2005]. Like emergence there is a number of different definitions across literature for self-organisation. The following does not aim to provide a new definition of self-organisation but instead to differentiate it from emergence and present a working definition in the context of this thesis.

Gershenson and Heylighen describe self-organisation as the process where global order is created from local interactions [Gershenson and Heylighen, 2003]. Similarly, Rocha defines it as the "formation of well organized structures, patterns, or behaviours, from random initial conditions" [Rocha, 1998]. In this sense, self-organisation can be thought of as the process through which macro-level patterns and mechanisms arise in the system, while emergence refers to the fact that the pattern or structure appeared without being represented at the micro-level of the system [Di Marzo Serugendo et al., 2006].

The novelty or irreducibility of the structure is not necessarily considered in self-organisation while it is a fundamental requirement for emergence. In addition, although micro and macro-levels of the system are at least implicit in the definitions, no reference is made to any causal link from the macro-level to the micro-level. To put this another way, the presence of downward causation is not necessary. De Wolf and Holvoet state that decentralised control across the agents is not essential for self-organisation, while also asserting that both self-organisation and emergence can occur independently of one another [De Wolf and Holvoet, 2005].

As with emergence, a lack of external control or input is a core requirement for a system to possess self-organisation. For example, De Wolf and Holvoet explicitly refer to this feature when describing self-organisation as "a dynamical and adaptive process where systems acquire and maintain structure themselves, without external control" [De Wolf and Holvoet, 2005]. Similarly, Di Marzo Serugendo et. al assert that self-organisation requires that the global system structure comes about without explicit control or constraint from outside the system [Di Marzo Serugendo et al., 2004].

For the purpose of this thesis, self-organisation refers to the ability of the system function without external control. Emergence refers to macro-level behaviour and properties arising out of the local actions and interactions of the agent, with a more detailed definition described in Chapter 3 (*cf.,* Section 3.2.2).

2.1.5 Summary

This section outlined the history of the concept of emergence and described the widely accepted features of emergence. These characteristics make emergence difficult to study as the emergent property and behaviour is irreducible to the properties of the constituent agents of the system. This irreducibility and novelty means that emergence is unpredictable, both in terms of when and how the emergence will arise. Emergence can be either harmful or beneficial to the overall system or the individual agents however, the absence of decentralised control and the non-linear behaviour and interactions of the agents mean detecting emergence is a challenge. After introducing emergence and its characteristics, the techniques used to study emergence were outlined, with four research fields giving rise to the science of complexity theory. After this, different categorisations of emergence were presented demonstrating the importance of downward causation and the identity of the observer in the context of emergence. With this background established, the work described in this thesis was positioned in the gap between Fromm's Type-II (mechanistic) and Type-III (reflective) emergence, where feedback from the macro-level influences the agents at the micro-level and the agents themselves act as observers.

In the next section, the systems that are the focus of this thesis, complex adaptive systems, are described along with computer simulation techniques used in their study. Particular emphasis is placed on multi-agent systems, which are used throughout the rest of this thesis.

2.2 Complex Adaptive Systems

This section introduces the systems that are the focus of this thesis, complex adaptive systems. It describes the features of these systems and illustrates how these characteristics allow emergence to occur. In addition, the computer simulation techniques used to model and study these systems are described, with emphasis placed on multi-agent systems, which are used in the rest of this thesis.

2.2.1 Introduction

Complex adaptive systems (CAS) are composed of interacting adaptive autonomous entities, called agents, that produce dynamic patterns and structures [Page, 2010, Dooley, 1996]. This evolution is in the form of adaptations with agents changing their properties or behaviours as they accumulate experience in the system [Gell-Mann, 1994]. A significant proportion of this adaptation effort is spent adapting to other adaptive agents in their environment. The interaction of these multiple adaptation mechanisms makes these systems complex and difficult to study [Holland, 1992]. In his seminal work on the field, Holland identified seven basic characteristics of CAS, consisting of four properties and three mechanisms [Holland, 1995]. These are:

- **Aggregation:** *property* Complex large-scale behaviours emerge from the interaction of less complex agents in the system. Control of these interactions is distributed with the system containing no-global controller [Arthur et al., 1997]. Such aggregates can act as "meta-agents" at higher levels in the system, possessing greater intelligence and flexibility compared to its individual constituents. An example provided is an ant nest, which is more adaptive and robust compared to individual ants.

- **Non-linearity:** *property* Agents interact in dynamic and non-linear ways, with reinforcement of chance events leading to an enormous number of alternative development pathways for the system [Levin, 1998]. These non-linear interactions have the effect of making the behaviour of the aggregate more complicated than would be expected if only the sum of all interactions was considered. Levin also points out that these interactions between agents are local [Levin, 1998].

- **Flows:** *property* Agents organise into networks of interaction in which one interaction may trigger other interactions. These flows of interaction can have a multiplicative effect when an agent inserts a new resource into the system, such as money, goods or a message. It is also possible for the effect to be a re-cycling effect, when the network contains a cycle.

- **Diversity:** *property* Agents evolve to fill diverse niches, which depend upon their interactions with other agents. An example provided is insects in a rainforest, which mimic their environment to avoid predators. In addition, diversity allows that if one type of agent is removed from the system, the system responds with a "cascade" of adaptations to fill the gap left.

- **Tagging:** *mechanism* Agents can be differentiated from one another based on the properties they possess. Gell-Mann refers to this characteristic as identification, where regularities can be differentiated from randomness in the environment [Gell-Mann, 1994]. This allows aggregation to occur because it facilitates selective interactions, providing a basis for filtering, specialization and cooperation. As a result, the tagging mechanism leads to hierarchical organisation in the system.

- **Internal Models:** *mechanism* Agents possess internal representations of themselves and the system, that allows them to exploit the regularity of their interactions with others. Gell-Mann refers to these internal models as "schema" which compress regularities and are improved through adaptation and evolution [Gell-Mann, 1994]. These representations enable the agent to anticipate, either implicitly or explicitly. An example is how a bacterium follows a chemical gradient, implicitly anticipating food lies in that direction,

- **Building Blocks:** *mechanism* Agents are presented with repetition, allowing internal models to be built. This repetition also allows for experience to be gained where the agent can, over time, decompose complex situations into parts. This decomposition allows the experience gained to be used elsewhere, in novel situations.

There is similarity in the terminology used to describe the properties and mechanism of CAS and the characteristics of emergence that were outlined in Section 2.1.1. This is unsurprising given that CAS are studied in the context of emergence however, it provides some clarity into how emergence arises in these systems. Examples of CAS are abundant and varied. They include: the internet [Rupert et al., 2008]; economies [Tesfatsion, 2003]; pedestrian crowds [Vizzari et al., 2013]; the nervous system [Koch and Laurent, 1999]; traffic systems [Wang, 2010]; and flocking behaviour [Niazi and Hussain, 2011].

2.2.2 Modelling CAS

Computer simulated models play a significant role in investigating CAS [Miller and Page, 2009]. These models reduce the system to its most essential parts which simplifies the system while simultaneously retaining the features of CAS such as non-linear interactions and aggregate behaviour. They additionally allow the system to be observed repeatedly in a controlled environment therefore facilitating detailed experimentation. Brownlee identifies a number of modelling techniques used by researchers in the field [Brownlee, 2007]. The following sections review the most common computer simulation techniques used for studying CAS.

2.2.2.1 Cellular Automata

Cellular automata (CA) are mathematical models of complex natural systems containing a large number of identical components that interact locally [Wolfram, 1984]. Both space and time are discrete, with space partitioned into distinct units called “cells” which can be in one of a finite number of states at any one time [Langton, 1990]. The state of any cell at time t is a function of its own state and the state of its immediate neighbours at time $t-1$. Each model of CA is deterministic with the initial state of all cells determining the future evolution of the system.

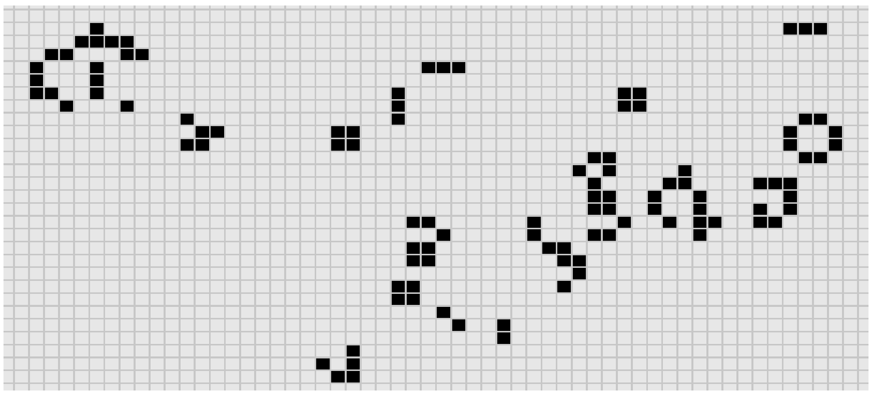

Fig. 2.1: ***Conway's Game of Life:*** *These mathematical models demonstrate the formation of coherent patterns from local interactions [Cornell, 2015].*

A famous example of CA is Conway's "Game of Life" [Gardner, 1970], illustrated in Figure 2.1, where coherent patterns such as "gliders" and "oscillators' seem to move across the system even though the individual cells are stationary. The rules are simple, with a cell turning on (alive) if the number of its direct neighbours is neither too small nor too large. Conversely, cells die if too few of their neighbours are on (loneliness) or if too many are on (overcrowding). CA have frequently been used to simulate CAS [Hoekstra et al., 2010]. Specific examples include the immune system's response to the AIDS virus [Grilo et al., 2002], communications systems [Goldenberg et al., 2001] and social systems [Miller and Page, 2009].

2.2.2.2 Artificial Neural Networks

Artificial neural networks (ANN) are a form of statistical learning models that are inspired by biological networks. Their structure facilitates adaptation to occur while the model is learning, with the ultimate goal of estimating a function given a set of inputs. They are composed of an interconnected group of nodes, called neurons, whose structure changes during the learning process. The network takes a set of inputs and through one or more hidden layers, computes a series of outputs. This is illustrated in Figure 2.2, with each circle representing a neuron and the lines between them representing weights. The strength of these connections fluctuates during the training phase as each new set of inputs generate certain outputs. Once the training is complete, these weights will determine the output that will be generated, representing the solution with the highest probability of success.

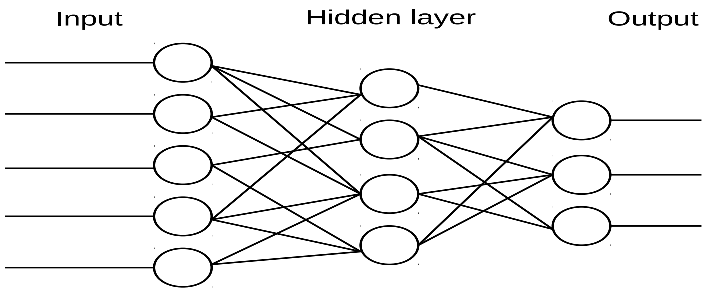

Fig. 2.2: ***Neural Network:*** *An example of a neural network with input, hidden layer and output.*

They have been applied to a variety of different problems such as handwriting recognition [LeCun et al., 1989], electricity load forecasting [Park et al., 1991] and in clinical medicine [Baxt, 1995]. The number of connections between neurons and the ability to have an unlimited number of hidden layers means that ANN's can produce complex global behaviour that emerges from the interactions of the neurons [Goh, 1995]. They have been used in the study and control of systems such as traffic management [Ledoux, 1997], organisational cybernetics [Alvarez-Molina et al., 2014], autonomous robots [Ramdane-Cherif, 2007] and the response of ecologies to climate change [Poff et al., 1996].

2.2.2.3 Learning Classifier Systems

Learning classifier systems (LCS) are a machine learning technique which combines reinforcement learning, genetic algorithms and other heuristics to produce adaptive systems [Bull, 2004]. Introduced by Holland [Holland, 1976, Holland, 1980], they are a rule-based system with mechanisms that allowed rules to be processed in parallel, where rules are tested and new rules are created through adaptation [Booker et al., 1989]. The conceptual architecture of Holland's LCS is illustrated in Figure 2.3, where the LCS receives input from its environment that are stored in a message list. A list of N condition-action rules make up the rule-base, and this is scanned when new input arrives to determine which conditions match the input. All matching rules are selected to create a match-set of length M, and a rule is selected from this match-set to become the system's action, based on a bidding mechanism. If a reward is generated by the environment

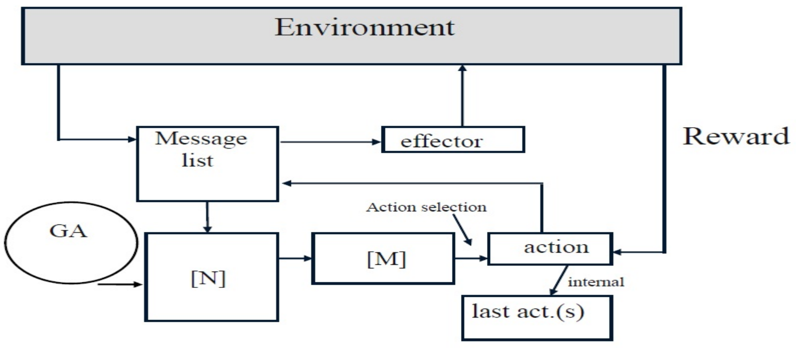

Fig. 2.3: ***Holland's Learning Classifier System:*** *The architecture of a learning classifier system [Bull, 2004].*

then it is assigned to the winning rule which generated the action. A steady state genetic algorithm (GA) is applied to the rule-set periodically where parent-rules are selected at random and new offspring are produced based on mutation and crossover. These new rules are inserted into the rule-set replacing existing rules.

LCS are used to solve problems that involve classification or can be constructed as a reinforcement learning problem [Whitehead and Ballard, 2014]. They can also be used in a predictive framework where the goal is to predict future environment states or properties [Woźniak et al., 2014]. Applications of LCS in CAS like systems include simulating the immune system and HIV in conjunction with cellular automata [Tay and Jhavar, 2005]. They have also been used to control traffic signals [Bull et al., 2004] and to devise stock trading strategies [Liao and Chen, 2001].

2.2.2.4 Agent-based Modelling and Simulation

Agent-based modelling and simulation (ABMS) [Kauffman, 1993, Holland, 1995] is an approach to modelling CAS composed of interacting autonomous agents. As the name suggests, agents are the foundation of this technique where a bottom-up approach allows self-organisation and emergence to be observed in these models. Each agent is self-contained and can be identified from other agents [Macal and North, 2010]. A set of relatively simple rules is used to describe each agent's behaviour and interactions with other agents. Agents in the model share an environment and it is also possible for interactions to occur between an agent and its environment. Agents can be goal-directed or have the ability to adapt its behaviour as the system evolves.

ABMS allow elements such as game theory [Bonabeau, 2002], sociology [Macy and Willer, 2002], evolutionary programming [Galván-López et al., 2014] and Monte Carlo methods [Ambrose and Grasela, 2000] to be incorporated allowing complex individuals such as humans to be modelled. This versatility and bottom-up approach has meant that ABMS have been used extensively in the study of CAS with the concept of modular interacting agents providing a neat abstraction of the composition of CAS [Jennings, 2001]. Examples of their application to CAS include the economy [Farmer and Foley, 2009], the Smart Grid [Taylor et al., 2014] and air transportation [Bouarfa et al., 2013]. ABMS has strong roots in the fields of multi-agent systems (MAS) and artificially intelligent robotics [Macal and North, 2005]. MAS is used through this thesis for simulation and evaluation and it is discussed in greater detail in the next section.

2.2.3 Multi-Agent Systems

A multi-agent system (MAS) is defined as "a loosely coupled network of agents that interact to solve problems beyond the individual capabilities or knowledge of each agent" [Sycara, 1998]. As with CAS, the foundational component of a MAS is an agent. An agent is defined as an "a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its delegated objectives" [Wooldridge, 2009]. These agents can also be "intelligent", referring to agents that pursue goals and execute actions that try to optimize given performance measures [Weiss, 1999].

Figure 2.4 provides an abstraction of the relationship between an agent and its environment, with the agent obtaining percepts of information through a set of sensors and acting on the environment through a set of effectors. The agent's environment includes other agents in the system and the ability to interact with other agents is a key capability. Interactions refer to the ability of agents to affect on another in pursuing their goals or executing tasks [Weiss, 1999]. This interaction can be in the form of direct communication, such as exchanging information in a shared language. Interactions can also occur indirectly through the environment, such as one agent observing another or stigmergy, where agents deposit a substance called a pheromone into the environment that can be sensed by other agents [Grassé, 1959, Dorigo et al., 2000].

Sycara identifies four characteristics of MAS. First, each agent has incomplete information or capabilities and thus a limited view of the system. Second, control is distributed across

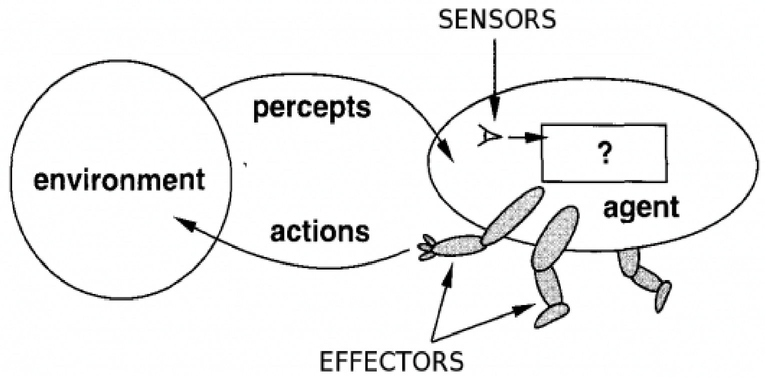

Fig. 2.4: ***MAS Agent:*** *The agent acquires information from its environment through sensors and produces actions as output that affect the environment [Russell et al., 1995].*

the constituent agents with no central global controller. Third, data are decentralised in the system. Finally, computation is asynchronous across the agents. These characteristics make MAS a suitable way of means of approximating CAS and for studying emergence and self-organisation [Holland, 2000, Dessalles et al., 2007, Noel and Zambonelli, 2014, Zambonelli et al., 2015, Lopez, 2015]. MAS have been used to simulate a variety of CAS exhibiting emergence such as flocking [Olfati-Saber, 2006], ant colonies [Xiang and Lee, 2008], swarm dynamics [Das et al., 2014], pedestrian counter-flows [Procházka and Olševičová, 2015], traffic management [Arel et al., 2010] and culture [Morris et al., 2014].

Bernon et al. describes five mechanisms used in research to generate self-organisation in MAS, where emergence may also exist as a consequence [Bernon et al., 2006, Di Marzo Serugendo et al., 2006]. These self-organising mechanisms include forms of interaction, direct or indirect, as well as engineering cooperation and reinforcement between agents to achieve a desired global behaviour. Fernandez-Marquez et al. present a catalogue of bio-inspired mechanisms in the form of design patterns to ease the engineering of artificial systems with self-organisation and emergence [Fernandez-Marquez et al., 2012]. These patterns are organised into layers with basic patterns such as spreading and aggregation forming the foundation for higher-level patterns such as flocking and quorum sensing.

Cooperation between agents provides the foundation for the Adaptive Multi-Agent-System (AMAS) theory [Gleizes et al., 1999], where a desired collective behaviour emerges as a result of cooperation between agents. Each agent in the system has the ability to adapt its interactions with other agents and the environment based on knowledge it has gained during the systems evolution, how it perceives other agents and the goal it seeks to achieve [Capera et al., 2003]. Agents possess local cooperation rules that enable it to detect and solve Non-Cooperative Situations (NCS), which indicate cooperation protocols have not been obeyed or that an unpredictable situation has arisen [Bernon et al., 2005]. The AMAS framework enables bottom-up engineering of systems to achieve a desired global function, however no attempt is made by the agents to detect when emergence has occurred so harnessing or controlling emergent behaviour is not addressed [Di Marzo Serugendo et al., 2006].

The Self-aware Pervasive Service Ecosystems (SAPERE) project [Sapere, 2015] provides a nature-inspired coordination framework to support the design and development of composite pervasive service systems [Zambonelli et al., 2011]. The approach positions adaptive capabilities in the overall self-organizing dynamics of the system rather than in individual components [Zambonelli et al., 2015,Montagna et al., 2012]. Its model is based on an assumption of spatially-based local interactions where agents can combine with each other to serve their own needs as well as those of the overall environment to create a "computational ecosystem". Interaction and coordination between agents are governed using eco-laws which define how agents can discover local or remote information and services as well as how information can be aggregated across agents. These laws form the foundation for a set of patterns that enable self-organisation and situational awareness across agents, however no attempt is made to detect unexpected emergent behaviour in the system [Zambonelli et al., 2015].

Paunovski et al. identify three properties common in MAS that exhibit emergence [Paunovski et al., 2009]. The first is that agents are mobile and can reposition themselves in the environment. Second, agents have the ability to influence the environment either through self-replication, stigmergy or by interacting with others. Finally, there exists the ability to distinguish between groups and individuals, for example a flock of birds or a lane of pedestrians. This is a similar observation to that made by Fromm who noted that in MAS with emergence we, the observer, can identify various levels of spatial resolution [Fromm, 2005].

This form of emergence is illustrated in Figure 2.5, where a MAS simulation of boid flocking is shown [Reynolds, 1987]. The coherent flocking behaviour apparent in the image on the right is generated through simple local interactions as agents move in the environment to both avoid each other and align their speed and heading. This spatial emergent behaviour has been studied

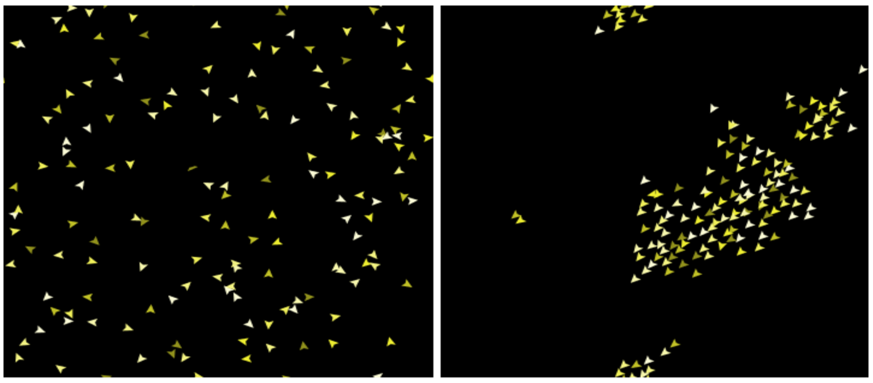

Fig. 2.5: ***Multi-Agent Flocking:*** *The image on the left demonstrates no emergent behaviour. The image on the right demonstrates emergent flocking behaviour.*

in a diverse set of MAS, such as the warehouse location problem [Chatty et al., 2013], pedestrian counter-flow [Procházka and Olševičová, 2015] and swarm behaviour [Minar et al., 1996] . MAS was chosen as the means of simulating emergence and evaluating the performance of the distributed algorithm, DETect, that is described in this thesis. Informed by other researchers in the field, emergence in the form of spatial behaviour and patterns is considered, as defined in greater detail in Chapter 3 (*cf.,* Section 3.2).

2.2.4 Summary

This section introduced systems composed of interacting adaptive autonomous agents known as complex adaptive systems (CAS). Emergence is a hallmark of CAS with dynamic patterns and structures appearing at the macro-level of the system as a result of interactions of the agents at the micro-level. The characteristics of these systems were described, which consist of four properties and three mechanisms. These characteristics are similar to those used to describe emergence, illustrating how emergence can form in these systems. Next, a variety of techniques for studying CAS using computer simulation were described. In particular, focus was placed on multi-agent systems (MAS) as a simulation tool, introducing its fundamental features and describing the type of emergence, spatial behaviour and patterns, that is typical in MAS. This background positions the research in this thesis, identifying the type of simulation and type of emergence that will be referred to in the rest of this thesis.

2.3 Detecting Emergence

Emergence can be both beneficial and harmful to the system and the constituent agents. As a result, detecting emergence when it occurs is desirable to enable actions to be taken to either leverage or mitigate the emergent behaviour or properties. In Section 1.3 the set of challenges presented by emergence detection in CAS were described, which included the decentralised nature of these systems and that emergence arises from unpredictable non-linear interactions between autonomous agents. In addition, the unpredictability of emergence means that knowing what to look for is not always obvious, while the dynamic nature of emergence means that detection should be timely to allow appropriate action to be taken. In this section, existing emergence detection techniques are presented and analysed with analysis informed by these challenges. The three categories of emergence detection techniques identified by Teo et al. are used to structure the analysis of the literature [Teo et al., 2013]. In each case, individual examples are described before the overall strengths and weaknesses of the approaches are discussed.

2.3.1 Variable based approaches

Variable based approaches employ system wide variables and statistical analysis to determine the existence and extent of emergence in a system. Such approaches are similar to the work described in this thesis and allow detection of emergence to occur at runtime. A major difference is that they assume that the global state of the system can be characterised by an observed variable or set of variables, with the presence of emergence concluded when these variables exhibit specified properties. Additionally, detection is done in a centralised fashion with a single, typically external, component undertaking the evaluation. These approaches can be sub-categorised into those that are based on entropy and those that are based on other types of system variables.

2.3.1.1 Entropy Based

The concept of entropy originated in the field of thermodynamics in the mid-19$^{\text{th}}$ century [History Of Entropy, 2015]. It was a response to observations that some energy released in combustion reactions is always lost to dissipation and therefore is not available for useful work. Entropy became the measure of this lost energy and is commonly understood as a measure of disorder in a system. In the field of information theory, Claude Shannon developed the concept of information entropy [Shannon, 1948], an analog of classical thermodynamic entropy, and the basis for the detection techniques discussed here. Shannon's entropy is described in equation 2.1, and it is a measure of the uncertainty or unpredictability of a system in respect to a certain variable, X,

$$H(X) := -\sum_{x \in X} Pr(x) log Pr(x) \tag{2.1}$$

Mnif and Muller-Schloer define emergence as the formation of order from disorder based on self organisation [Mnif and Muller-Schloer, 2006]. This order relates to the coherent behaviour, structure or property that represents emergence at the macro-level of the system. The authors propose that emergence can therefore be quantified as a function of the system entropy, measured for a number of system specific attributes, from the beginning of its process and at its end. The measurement requires some level of abstraction in the observation model that will automatically lead to a lower entropy value even when there is no change in the system. Emergence is therefore the unexplained difference in entropy in the system once this change in abstraction level is accounted for. The approach is applied in a simulation of bird swarms with states containing clusters of birds demonstrating a higher degree of order in their positions compared to when no clustering was present.

Fisch et al. propose an extension to Mnif and Muller-Schloer, enabling multiple continuous and hybrid system variables to be used in the entropy calculation [Fisch et al., 2010]. Continuous variables are incorporated using density functions with Hellinger divergence (Equation 2.2) used to measure the difference between historic, p, and current, q, density functions of the same variable, x. These variables are assumed to be output from a process under observation. This approach gives a measure of the degree of emergence on a scale between 0 and 1, with values close to 0 indicating no emergence in the system.

$$Hel(p, q) = \sqrt{1 - \int \sqrt{p(x)q(x)} dx} \tag{2.2}$$

Procházka and Olševičová focus on self-organisation of pedestrians into emergent counter-flows in a qualitative microscopic pedestrian simulation [Procházka and Olševičová, 2015]. The authors observe the difference in entropy in the system when these emergents are present compared to periods when agents undertake a random walk. Information entropy is calculated as a function of the number of agents in environment segments and whether they are currently in a lane or not. A lane is deemed to exist if a certain threshold of agents are moving in the same direction with a maximum distance and offset angle between them, and no other agents moving in the opposite direction exist between them. The entropy calculation is updated periodically as each simulation runs, with the system entropy dropping during periods of counter-flow compared to periods of random walks.

Shalizi and Moore detect emergence by the ratio between the statistical complexity of the system and the predictability of the macro-level system state [Shalizi and Moore, 2003]. This is done by characterising emergence as the relationship between two sets of variables where a) one set of variables is a coarse-graining of the other and b) the coarse-grained variables can be predicted more efficiently than the fine-grained. Thus in a system, the coarse grained variables correspond to those variables that describe the macro-level of the system, with the second set describing the micro-level.

Holzer outlines a mathematical model for discrete complex systems that uses information entropy to define the level of autonomy and emergence in the system [Holzer et al., 2008]. This technique uses a directed graph to model interactions between agents in the system with each vertex corresponding to an agent and each edge corresponding to an interaction. To evaluate emergence, the level of information of all edges is compared to the level of information contained in each single edge. The ratio between these two information measures gives a measure of emergence in the system. The feasibility of the approach is demonstrated in the context of self-organised slot-synchronization in wireless networks.

2.3.1.2 Non-Entropy Based

Seth [Seth, 2008] uses a combination of linear and non-linear time series analysis, based on Granger Causality (G-causality) [Granger, 1969] to achieve emergence detection. This technique is inspired by Bedau's contention that emergence is both caused by and autonomous from the underlying agents of the system. The author defines a macroscopic property, such as the centre of a flock of birds, to be emergent when it is both statistically autonomous of, and statistically caused by, the microscopic properties (individual position of each bird). According to G-causality, a variable Y causes X, if including historic values of both Y and X in a linear regression model, reduces the prediction error of future values of X compared to just using historic values of X. A variable, X is said to be autonomous if including its own historic values in the model reduces prediction error compared to using a model composed of past states of other "external" variables.

Niazi and Hussain present the SECAS framework, which uses a distributed set of sensors to collectively sense the complex behaviour [Niazi and Hussain, 2011]. Their hypothesis is that emergent behaviour can be perceived through an environmental change by the sensors. These values can be used in an aggregation function to reflect the emergent effect as a manifestation of the global environmental change. Using boid flocking as an example scenario, the sensors are equipped with a proximity sensor that indicates if an agent is nearby at time t. The expected emergent behaviour, flocking, means that the agents will get close to one another and, as a result, the number of sensors detecting agents nearby will reduce. The sensors proximity status is counted centrally with the expected emergent behaviour said to have occurred when the number of sensors with agents nearby is consistently lower than the total number of sensors.

Grossman et al. present “Angle”, a hierarchical framework to detect emergent and anomalous behaviour in distributed clustered systems [Grossman et al., 2009]. The framework's architecture contains three types of nodes; sensor nodes which collect IP data, cloud nodes which run cloud based services and grid nodes which are pools of nodes running grid services. The detection is undertaken by the sensor nodes which monitor events occurring across temporal windows. The events for each window are collected and a feature vector is computed creating a model of that window. This model is compared to previous models with emergence being defined to occur when a model represents an outlier from previous models.

De Wolf et al. present a hybrid method for providing guarantees about system-wide behaviour in decentralised autonomic computing systems [De Wolf et al., 2005]. Their approach allows users to observe the evolution of macro-level properties when accurate models of micro-level properties are provided. It requires users to have knowledge of relevant variables in advance and that system-wide variables are obtainable.

Chan outlines an approach to emergence detection that is inspired by the fundamental importance of agent interactions in creating emergence in any system [Chan, 2011]. The framework is designed for agent-based simulations with detection facilitated by globally monitoring the number of interactions and state changes across all agents in the system. This creates a time-series metric of interactions with emergence said to exist when this interaction metric departs from normality. The author demonstrates the approach using three simulation models, game of life, boid flocking and brownian motion.

Birdsey and Szabo propose an architecture for identifying emergent behaviour in a multi-agent system as it occurs [Birdsey and Szabo, 2014]. The approach contains three stages; modelling, metric collection and, analysis and visualisation. The model specifies the system and the agents, while a set of metrics are collected and aggregated centrally at regular intervals during simulations. During analysis, the collected metrics are compared against either previous known system states that exhibited emergence, or a set of specific threshold values identified by a system expert. This approach allows for different metrics to be used depending on the specific system under study. The authors demonstrate the feasibility of the approach using boids flocking, with the specific metric, agent interactions, being modelled as a graph where each node represents an agent and a weighted edge represents the strength of interaction between two agents. Using the Hausdorff distance [Huttenlocher et al., 1993], a metric that is used to measure how similar two graphs are, simulations states containing visual flocking behaviour were shown to be closer to baseline states with flocking behaviour, identified by a system expert, compared to simulations states without visual flocking.

2.3.1.3 Summary

Variable Based Approaches characterise emergence using one or more system variables and use statistical and information theory techniques to determine when emergence has occurred in the system under observation. Information entropy is a common metric used to indicate emergence in these techniques (see Section 2.3.1.1). Variable-based approaches allow emergence detection to occur at runtime, assuming that the relevant system variables have been identified and can be observed.

A common drawback to each of these approaches however, is that they depend on knowledge of the global system state to calculate these measures. This is achieved using an omniscient centralised controller, violating the decentralised nature of CAS, and effectively means that detection must take place outside the system. This is illustrated by Seth's approach which, for a boid flocking model, requires the individual position of each agent and the centre of the flock to be known [Seth, 2008]. Entropy based approaches require a similar level information to calculate the information entropy of the entire system.

Niazi and Hussain and Grossman et al. use a more distributed architecture with the information collected by a collection of static "lookout" sensors [Niazi and Hussain, 2011,Grossman et al., 2009]. However, as with the other approaches described here, the processing of the collected data is done using some central sync-node. These sensors can be considered as an intermediary between the micro-level and the central controller and allow for the possibility that the constituent agents of the system could be informed of the emergence once it has been detected.

Another limitation of these approaches is that the specific variables needed for analysis must be known in advance. This assumes that these variables can be known and, more fundamentally, that the specific emergent behaviour or properties are predictable. However, as discussed in Section 2.1, this may not always be possible given the unpredictable and non-linear nature of both emergence and CAS. The most generic metric used is the interaction metric suggested by Chan [Chan, 2011] as it can theoretically be applied across different systems, however it requires that what constitutes an interaction is specified for each system. Moreover, it requires extensive knowledge of all aspects of the system and its constituent agents to compute.

2.3.2 Formal approaches

A second category of techniques use formal language and modelling approaches to facilitate detection. These approaches can be sub-categorised into those that are intended for design time use, to predict the possibility of emergence, and those that are intended for run time to detect if emergence has occurred.

2.3.2.1 Design Time Verification

Kubik provides an early example of this design time verification inspired by the notion that emergence is greater than the sum of the parts [Kubík, 2003]. The author uses a formal grammar to characterise basic (weak) emergence in multi-agent systems (MAS), with each agent's behaviour being defined by a language. This specifies what actions an agent can take, what states it can enter and what effect it can have on the environment. A superimposition over all permutations of each agent's language gives a sum of agent behaviour. At the same time, the entire MAS is modelled using a modified cooperating grammar. Emergence is said to be possible if the system as a whole can generate a language (behaviour) that cannot be generated by the superimposition of individual agent's languages.

Teo et al. present a similar approach that addresses a limitation of Kubik [Teo et al., 2013]. In Kubik [Kubík, 2003] the superimposition step across all agent languages leads to state-space explosion and causes the technique to be prohibitively expensive except for rudimentary systems. Teo et al. address this by limiting analysis to only those agent behaviours and states that are possible and are of interest. This greatly reduces the computational work involved but assumes that the possible and interesting behaviours can be known in advance. Szabo and Teo [Szabo and Teo, 2015] describe an extension to this approach where the degree of interactions between agents is used to identify the states of interest. This interaction metric is similar to the concept outlined by Chan [Chan, 2011], and means that all details of the emergence is not required allowing the technique to be used on different systems such as flocking, Game of Life and traffic jams. Once a system has been simulated, the system state at each time step is retrospectively analysed to determine if a sufficient degree of interaction occurred at that state based on a predefined threshold. If so, the state is added to the set of states that will have a superimposition calculated. This approach enables the time steps when emergence occurred to be identified allowing the underlying cause of the emergence to be investigated.

Moshirpour et al. present a model-based technique with the goal of detecting emergent behaviour at system design-time [Moshirpour et al., 2012]. Detection at this stage can be used to determine the cause of the behaviour, and thus eliminate or control the emergence. Here, emergent behaviour is characterised by implied scenarios, which are types of behaviour that are present in the synthesised model of the system but are not explicitly defined in its specification as a scenario.

Paunovski et al. present a framework for exploring emergence in complex systems using multi-agent simulations [Paunovski et al., 2008]. This framework involves two phases, beginning with a formal design and verification of the system model with respect to some expected behaviour. The second phase involves experimentation through model simulation where iterative "bottom-up" and "top-down" analysis is applied to detect interaction patterns, local properties and other elements that may cause or influence emergence. The authors discuss their framework in the context of emergent herd dynamics, suggesting that by working with a field expert, such as a biologist, system variables like herd cohesion could be identified as suitable indicators of the presence of emergence in the system under study.

2.3.2.2 Runtime Detection

Randles et al. advocates the use of a formal method to specify component interactions, system evolution and runtime global states [Randles et al., 2007]. This formal specification is achieved using Specification Language for Agent-Based Systems (SLABS) [Zhu, 2001]. This approach requires that known candidate scenarios of emergence be pre-defined through design time formalisms. At run time, an observer system is implemented with logical reasoning capabilities to determine if any of the pre-defined scenarios has arisen.

De Angelis and Di Marzo Serugendo present a logic language used to verify macro spatial properties of a self-organising system at run time [De Angelis and Di Marzo Serugendo, 2015]. Logic formulae, defined using language operators, depict the intended or expected emergent properties that arise through local interaction. An agent in the system can check if this emergent property exists using a chemically inspired coordination mechanism that involves agents communicating indirectly across a tuple-space. Each tuple contains passive data, such as its name and value, and a logic fragment that define how the message should be spread, merged and reacted to. The emergent property is verified by using a formula that is decomposed into sub-formulae, which are evaluated distributedly as the tuple traverses the system. Their results are recombined according to the meaning of the involved spatial operators. Once the tuple has traversed the system, if no agent's state violates the logic formulae specified by the originating agent, the emergent property is concluded to exist.

Ronald et al. proposed a subjective test for emergence based on the degree of novelty of the phenomenon [Ronald et al., 1999]. The test assumes that the system in question is designed using a formal language $L1$. An observer of the system uses a different formal language $L2$, but is also fully aware of the system design and of $L1$. Given this, emergence would occur if the causal link between what was happening in the system, described in $L2$, and its underlying design, $L1$, was non-obvious to the observer. Thus, this test of emergence can be summarised as *design, observation, surprise.* This is an early attempt at describing a test for emergence and although the authors do not specify details of how it can be implemented, it provides a general structure on which subsequent tests have been based.

Ciancia et al. describe a modal logic methodology for verifying macroscopic properties of a system that depends on space [Ciancia et al., 2014]. The authors define a spatial logic that can be used to express properties of closure spaces, a generalisation of topological spaces. This logic consists of two spatial operators, a one step modality that turns closure into a logical operator and a binary *until* operator which is interpreted spatially. This allows areas to be identified in a spatial model where inside the area some property, P, holds and simultaneously the area is surrounded by points where a different property, R, holds. The approach is not explicitly applied to emergence detection, with a case study showing how areas of interest on digital images and maps can be identified. Nonetheless, it is included here as the methodology could in principal be applied to systems where spatial emergent behaviour, such as flocking, is present. However, the authors concede that incorporating temporal reasoning into the methodology, in order to observe a system as it evolves, is likely to lead to state space explosion, with snapshot models being computed at every state.

2.3.2.3 Summary

Formal approaches require detailed modelling of systems and their components using formal language techniques. Techniques, such as Kubik, Teo et al., Moshirpour et al. and Szabo and Teo, are not intended to detect emergence at runtime [Kubík, 2003, Teo et al., 2013, Moshirpour et al., 2012, Szabo and Teo, 2015]. Instead, these approaches consider the possibility of emergence arising in systems and attempt to identify and study the underlying causes at design time, using simulation. These approaches are computationally expensive with the use of superimposition across agents leading to state-space explosion. State-space explosion is addressed both by Teo et al and Szabo and Teo, where the approach is improved by limiting analysis to only those states that are possible and of interest. However, this analysis occurs retroactively once the system simulation has completed, so discovering emergence cannot be used within the system itself. As a result, their applicability to CAS, where fundamentally unpredictable emergence may arise, is limited.

The second category of formal approaches enable detection to occur at runtime. Both Randle et al. and De Angelis and Di Marzo Serugendo. are limited by requiring that the specific emergent phenomenon or configuration of the system can be known at design-time or formally defined [Randles et al., 2007, De Angelis and Di Marzo Serugendo, 2015]. De Angelis and Di Marzo Serugend do provide a distributed mechanism with the constituent agents of the system coordinating using a tuple-space to achieve detection. However, determining the existence of emergence is done by one of the agents with the requirement that the original tuple traverses the entire system before this can be concluded. This requires that the system is effectively static while this processing takes place with the dynamic nature of emergence, formation and evaporation, not addressed. It is possible that the methodology described by Ciancia et al. could be used to allow detection and evaporation of emergence, however its centralised architecture and computational expense means that this is not practical in CAS [Ciancia et al., 2014].

2.3.3 Event-based approaches

The final category of detection methods can be described as event-based. These are a hybrid of the above approaches, incorporating aspects from both statistical analysis and formal approaches. As a result, they require detailed modelling of the system at design time but facilitate detection to occur when the system is executing.

Chen et al. outline a method for describing emergent behaviour at different levels of abstraction based on defining event types [Chen, Chih-Chun and Nagl, Sylvia B and Clack, 2008]. The authors differentiate between simple and complex events, with simple events formalised as agent state transitions when viewed from a particular level of abstraction. Complex events are a configuration of simple events where the configuration includes dimensions such as space or time. Once all events are formally described, statistical analysis is used to discover correlations between event types in simulations, enabling identification of relationships across abstraction levels of the system.

Lewis and Whitehead describe the Mayet Architecture, which is designed to detect emergent unwanted behaviour in games at runtime [Lewis and Whitehead, 2011]. This framework requires that the system designer specify a set of constraints defining expected system behaviour. At run time the system is monitored to determine if its state conforms to these designer constraints. Emergence is concluded to have occurred if a state is entered that violates these constraints. This definition of emergence is markedly different from what is considered emergence throughout the rest of the literature discussed here. The approach was demonstrated using a version of Super Mario World, where Mayet was attached and a taxonomy of faults were added to the game. The case study showed that Mayet could identify when an “emergent” state occurred and allowed steps to be taken to repair the fault at run time so that the game could continue.

2.3.3.1 Summary

Event-based approaches are a hybrid of formal and variable-based approaches. Similar to those categories, the approaches here rely on centralised architectures and require a significant level of knowledge at design time. For example, the framework described in Chen et al. is intended as a design time technique, requiring all event types in the system to be formalised [Chen, Chih-Chun and Nagl, Sylvia B and Clack, 2008]. Once this is done, emergence can be predicted in a system using simulation allowing measures to be possibly built into a system to avoid harmful emergence. Emergence however, is unpredictable and, as noted by Goldstein, even when it has been seen once, future evolutions will be different [Goldstein, 1999].

In contrast, the Mayet Architecture [Lewis and Whitehead, 2011] provides support for run-time detection of emergent events, relying on a centralised architecture where the observer has access to global system state. It assumes that the designer of the game has accurately and exhaustively identified all system constraints, which is possible for the systems being monitored, video games. However, achieving this level of completeness for CAS is more challenging due to both increased scale and their non-linear nature. Finally, their definition of emergence, as a state that violates these constraints, is a significant departure from the traditional view of emergence used throughout the literature.

2.3.4 Analysis

The existing emergence detection techniques, across all three categories, are evaluated against the challenges of emergence detection, with Figure 2.6 presenting this analysis as a KIVIAT (radar) diagram. Each criteria has 3 levels which increase in applicability for emergence detection in CAS like systems. The further away a detection category is from the center of graph the greater their applicability is for that challenge. The diagram illustrates the strengths and weaknesses of each detection category in the context of these challenges. The analysis highlights:

- In available literature, no fully decentralised emergence detection approach exists. All existing techniques require a centralised component to process system information, that may or may not be gathered in a distributed fashion.
- Existing techniques do not handle detection when the emergence is not known about and understood in advance. They require designers of the system to have knowledge of what

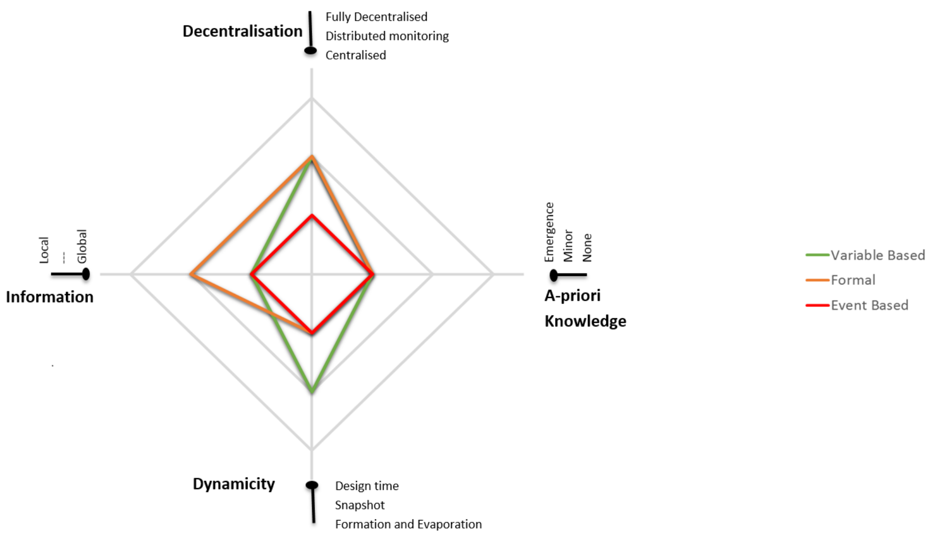

Fig. 2.6: ***State of the Art Evaluation:*** *The KIVIAT diagram shows how existing techniques address each of the four challenges of emergence detection in CAS. Each criterion has three levels which increase in applicability for CAS, the further they are away from the centre.*

properties or variables will indicate emergence exists or what system configurations qualify as emergence.

- Existing runtime approaches can determine if emergence is present in the system. However, no approach is capable of determining when the macro-state of the system is transitioning into and out of an emergent state. This limits the utility of these approaches as it may be too late to take appropriate action by the time the detection has occurred.

In summary, existing techniques depend on a centralised architecture and knowledge of the expected emergent phenomena at design time. Moreover, the dynamic nature of emergence as it both, forms and evaporates, in the system is not considered.

2.4 Summary

This chapter presented a detailed discussion of the concept of emergence, highlighting its commonly accepted characteristics and the different types of emergence that can occur. This analysis highlighted that emergence detection can be achieved by the constituent agents of a CAS at runtime as feedback from downward causation impacts the agents in the system. The leap from Type-II (mechanistic) to Type-III (reflective) emergence was identified as the position for the research described in this thesis. Bridging this gap allows the agents to act as detectors of emergence without an external or central observer and allows the effects of emergence to be either mitigated or leveraged.

The subsequent section introduced Complex Adaptive Systems (CAS), which are systems composed of autonomous adaptive agents that interact and these interactions result in emergence in the system. Computer simulations are a well established means of studying CAS using techniques such as cellular automata, artificial neural networks and genetic algorithms. This thesis uses multi-agent systems to simulate CAS and emergence. In multi-agent systems with mobile agents, emergence occurs in the form of coherent spatial behaviours and patterns, such as flocking.

The final section presented existing techniques for detecting emergence. These techniques can be classified into three groups, variable based, formal and event based. Variable based approaches use system wide variables and statistical analysis to determine the existence of emergence, allowing detection to occur at runtime. Formal approaches use a combination of formal grammars and simulation to both predict and detect if specified emergent behaviour occurs in the system. Finally, event based approaches are a hybrid of variable based approaches and formal approaches. All approaches were analysed in the context of the challenges presented by detecting emergence in CAS that were identified in Chapter 1 (*cf.,* Section 1.3). This analysis highlighted that existing approaches are limited by a centralised architectures and required a priori of the expected emergent phenomena at design-time. In the next chapter, the design objectives necessary to meet the challenges presented by emergence detection are outlined and the design of a novel distributed algorithm to meet these requirements is described.

Chapter 3

Design

This chapter returns to characteristics of emergence and the challenges associated with detecting it in Complex Adaptive Systems (CAS) to frame both the problem and solution that are the subject of this thesis. The chapter begins by identifying a set of design objectives that are necessary to achieve effective emergence detection (*cf.,* Section 3.1). Next, the terminology associated with both emergence and CAS are outlined to define a system model that scopes the contribution of this thesis (*cf.,* Section 3.2). The subsequent section presents a set of design decisions to address each design objective (*cf.,* Section 3.3), before DETect, the solution proposed by this thesis, is described (*cf.,* Section 3.4). This section is divided into three parts: The first examines how an autonomic model of the environment can be defined. The second part examines how feedback from downward causation can be detected. The third part examines how collaboration between agents can be facilitated to enable information to be shared and aggregated into larger views. Finally, the chapter concludes with a summary of the key design decisions and describes how the proposed solution achieves the design objectives outlined.

3.1 Design Objectives

Chapter 1 introduced the challenges presented by emergence detection in CAS (*cf.,* Section 1.3). CAS are composed of decentralised autonomous agents that interact in dynamic and non-linear ways. Macro-level properties and behaviours emerge from these interactions, that are unpredictable and cannot be reduced to the properties of the constituent agents. These emergents may be harmful or beneficial to the system or the individual agents, so it is desirable to detect emergence when it occurs. However, no agent in the system has a global system view and the unpredictable nature of emergence means that it is not always possible to know what form the emergence will take. Moreover, emergence is dynamic, it forms at some point during the system evolution and evaporates at some future point. These characteristics correspond to the challenges identified in Chapter 1 and motivate a set of design objectives for effective emergence detection in CAS. An effective emergence detection mechanism must:

- **Design Objective 1: Decentralise detection**

  Holland describes a CAS as being composed of "many distributed, interacting parts, with little or nothing in the way of a central control" [Holland, 1992] (challenge 1). This is consistent with the characteristics of emergence outlined by De Wolf and Holvoet [De Wolf and Holvoet, 2005] and necessitates that any effective detection mechanism should be similarly decentralised.

- **Design Objective 2: Detect emergence at runtime**

  Emergence is unpredictable in the way and form it can arise [Mogul, 2006,Goldstein, 1999] (challenge 3). Even when emergence is engineered into the system, undesired emergent behaviour can still arise [Di Marzo Serugendo et al., 2006]. This is compounded by the fact that CAS are composed of autonomous agents that behave and interact in dynamic non-linear ways (challenge 2). Therefore, effective detection of emergence must occur at runtime as the unpredictability of both emergence and CAS means that the effectiveness of design time approaches are necessarily limited. Additionally, by detecting emergence at runtime, the possibility of reacting to either leverage or mitigate the effects is presented.

- **Design Objective 3: Rely on locally available information**

  The decentralised nature of these systems means that no single agent or component of the system possesses a global view of system state (challenge 1). Each agent has access only to information from their locality, referring to the subset of the environment they can sense, including other agents that they interact and communicate with. Therefore, emergence detection must be facilitated using each agent's limited system view as a foundation.

- **Design Objective 4: Support collaboration between agents to achieve a larger system view**

  The fact that each individual agent has access to only a limited view of the system has already been identified (challenge 1). This presents a problem when we consider that detecting emergence entails detecting a property or behaviour that occurs at the macro-level of the system. As a result, no single agent is capable of concluding the existence alone, and it is necessary that they combine their individual views of the system to create an aggregate view with sufficient scope. This collaboration should be light weight with consideration given to the dynamic and non-linear ways that agents may interact (challenge 2).

- **Design Objective 5: Autonomously select data to monitor**

  Emergence is inherently unpredictable, even when it has appeared once in a system, future evolutions will involve some alterations [Goldstein, 1999]. As a result, knowing the exact characteristics or properties to monitor becomes a challenge (challenge 3). An emergence detection mechanism could address this by monitoring all variables an agent has access to, however, with no upper limit on the number of variables that an agent or system could have, this is impractical. Instead, the properties of the system to monitor should be determined at runtime and be selected from the set of variables already available to agents.

- **Design Objective 6: Detect formation and evaporation of emergence**

  Emergence is dynamic, capable of both forming and evaporating as the system evolves in time (challenge 4). This means that for a detection mechanism to be useful, it must consider this aspect as its utility will be diminished if it detects an emergent phenomenon after it has already evaporated. Instead, the maximum utility is achieved if the emergent behaviour can be detected while it is still forming or evaporating. Detecting these transition periods provides the most time for an interested party to react before the macro-state becomes settled and therefore mitigate or leverage the effects of the emergence.

Collectively these objectives target the challenges presented by emergence detection in CAS, with Figure 3.1 illustrating the challenge that motivates each objective. The objectives highlight the tension between the goal of detecting an unpredictable behaviour or property that is global in

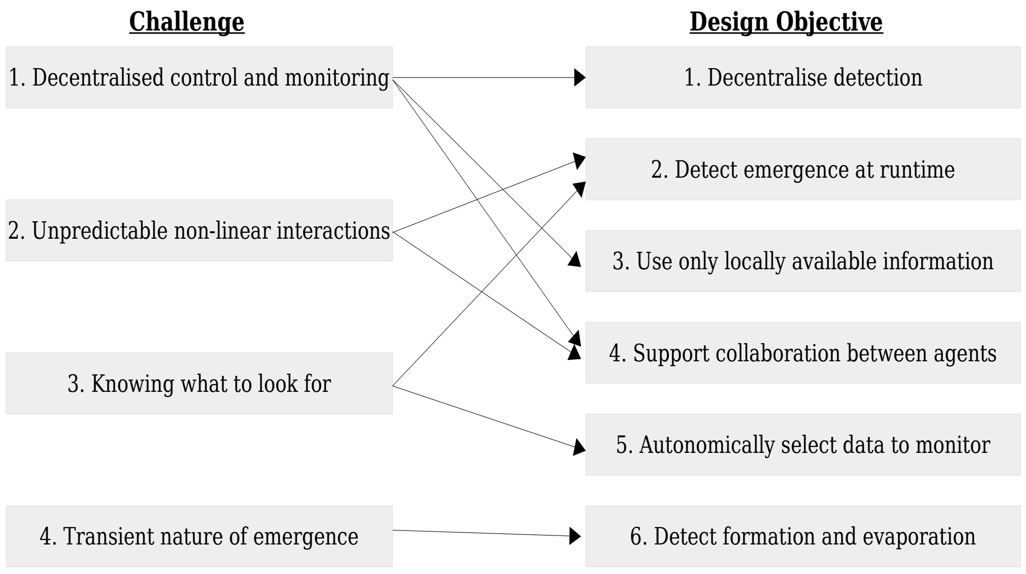
Challenge
1. Decentralised control and monitoring
2. Unpredictable non-linear interactions
3. Knowing what to look for
4. Transient nature of emergence
Design Objective
1. Decentralise detection
2. Detect emergence at runtime
3. Use only locally available information
4. Support collaboration between agents
5. Autonomically select data to monitor
6. Detect formation and evaporation

Fig. 3.1: ***Design objectives*** *to address emergence detection challenges.*

nature using only locally available information. The remainder of this chapter presents DETect and analyses how its design addresses the above objectives.

3.2 System Model

This thesis presents DETect, a novel distributed algorithm that facilitates decentralised detection of emergence in CAS. In this section, the terminology associated with both emergence and CAS as it relates to this thesis is defined. In this way, the type of systems that are the focus of this research are clarified and the scope of, and assumptions made by, the design of DETect are outlined.

3.2.1 Terminology

The systems that are the focus of this thesis are Complex Adaptive Systems in the context of multi-agent systems.

Definition 1. *A Complex Adaptive System is a decentralised system composed of autonomous, adaptive, interacting agents, where agent interactions can potentially give rise to emergent behaviour.*

Definition 2. *A Multi-Agent System is a loosely coupled network of mobile agents that interact in a shared environment.*

Definition 3. *An agent is a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its own objectives.*

The definition of Complex Adaptive Systems is consistent with the generally accepted definition found across literature [Holland, 1992, Page, 2010]. The definition of a multi-agent system is similar to that outlined by Sycara [Sycara, 1998], with the definition of an agent taken from Wooldridge [Wooldridge, 2009]. Unlike Sycara however, the definition of multi-agent systems here is limited to those systems composed of mobile agents and no assumption is made regarding the purpose of agent interactions. These definitions require that an additional set of definitions are presented to outline what is meant by *autonomous, interaction and adaptation.*

Definition 4. *Autonomous refers to the property of each agent where their behaviour depends on their own experience [Weiss, 1999].*

Definition 5. *An interaction refers to an instance where one agent influences the behaviour or decision making of another agent. Such interactions can be direct, meaning an intentional, explicit exchange of messages between agents. Interactions may also be indirect, such as an agent observing another agent.*

Definition 6. *Adaptation refers to an agent's ability to change its behavioural rules at runtime.*

This definition of adaptation is consistent with that used by authors such as Holland and Gell-Mann in the context of CAS [Gell-Mann, 1994, Holland, 1992]. The position of DETect in the context of an adaptation pattern is discussed in Section 3.3.2 below.

3.2.2 Emergence

Section 2.1.3 described different categorisations of emergence based on the extent and complexity of downward causation in the system and the role and observer of the emergence. This thesis focuses on emergence that can be harmful or beneficial to the constituent parts of the system. Therefore, emergence where no downward causation exists is excluded from consideration. Instead, the transition between Fromm's Type II and Type III emergence [Fromm, 2005], or De Hann's mechanistic and reflective emergence [De Haan, 2006], is where the work in this thesis is positioned. Additionally, as the systems described in this thesis are MAS composed of mobile agents, emergence refers to macro-level spatial patterns and behaviours that are generated by the location interaction of agents (*cf.,* Section 2.2.3). Examples of such emergents are flocking, traffic-jams, pedestrian lane formations. Therefore, emergence is defined as:

Definition 7. *Emergence is the appearance of coherent spatial patterns and behaviours at the macro-level of the system that are caused by the interactions of agents at the micro-level but are not specified in the agents behavioural rules. This emergence influences the agents at the micro-level through downward causation.*

Definition 8. *An emergent event refers to a transition of the system's macro-level into or out of a state of emergence. That is, the period of time during which the emergence is forming or evaporating.*

3.2.3 Assumptions

A DETect-based system consists of the following components and properties:

- A set of agents A $=\{A_1,...,A_n\}$, where each agent controls a set of sensors and actuators.
- Each agent A_i contains a set of internal variables $IV_i=\{IV_{i1},...,IV_{ip}\}$, that the agent uses to describe itself.
- Each agent A_i contains a set of external variables $EV_i=\{EV_{i1},...,EV_{ir}\}$, that the agent uses to describe its environment.
- The set of internal and external variables for each agent A_i does not change during the system evolution.

- System time is synchronised across all agents, with agents updating internal and external variables and making decisions simultaneously at fixed time intervals.
- Each agent A_i has access to a set of neighbours $N_{it} = \{N_{it1},...,N_{itj}\}$ consisting of all other agents $A_j \in A$ that are one-hop neighbours of the agent A_i at time step t. One-hop refers to the agents ability to directly communicate with that agent at that time step t.
- Each agent contains an *ActionSelector* or *AdaptationManager* which can receive the emergence detection event from DETect and select an appropriate action in response.
- Agents are failure free, meaning agents can always contact their one-hop neighbours and incomplete or inaccurate information is not communicated.

3.3 Design Decisions

Muller suggests that emergence can be detected by special "lookout" agents in a system if the agents know what to look for and they have a sufficiently global view of the system [Muller, 2004]. Although, such lookout agents are not applicable in the systems discussed in this thesis (challenge 1 and challenge 3), this assertion crystallises the problem of detecting emergence into two components; knowing what to look for and having the appropriate scope. The following discussion outlines the design decisions made to achieve these two components and shows how these decisions address the design objectives that were identified in Section 3.1.

3.3.1 Decentralised Detection

Control and monitoring are decentralised in CAS, distributed across the constituent agents that compose the system. The interactions of these agents generates the emergence with the effects of emergence feeding back onto the agents through downward causation. This gives agents a stake in detecting emergence as any detection can potentially be used to improve the effects the emergence has on them. At the same time, as the generators of the emergence, agents are best placed to potentially influence it by altering their behaviour or interactions. With this in mind, the first design decision is as follows:

> **Design Decision** 1 **- Constituent agents act as detectors** The first design decision is the most fundamental, identifying the constituent agents as

the detectors of emergence. Agents both create and are influenced by emergence at run time meaning that they are, in principle, capable of detecting and shaping it. This makes them the ideal candidates to act as detectors without the need for any special "lookout" agents or a centralised monitor. Therefore, this design decision addresses both design objective 1 and 2.

3.3.2 Detection in the context of adaptation

The Monitor, Analyse, Plan and Execute (MAPE) loop [Kephart and Chess, 2003] is an extensively used model for adaptive systems management. Although it did not originate in the MAS domain, it is an example of a generic autonomic feedback loop and could in principle be used to structure significant adaptations to agent behaviour [Cabri et al., 2011]. This is used as a model of an adaptation control pattern to position the work described in this thesis.

The MAPE loop has four phases:

- **Monitor -** which collects the details from the managed resource or system. The monitor phase also aggregates and filters these details until a symptom requiring further analysis is identified.
- **Analyse -** which performs complex data analysis and reasoning on the symptoms described by the monitoring function. If a change is required, a change request is passed to the planning phase.
- **Plan -** which creates or selects a procedure to make a desired alteration to a managed resource. In a MAS context, this could include the agent's behavioural rules.
- **Execute -** which changes the behaviour of the managed resource using effectors, based on the actions recommended by the Plan phase.

Design Decision 2 **- DETect is in the monitoring phase of a MAPE loop** The monitoring phase of an adaptation control pattern is where the information from the system is managed and initial symptom discovery occurs. This phase is the most suitable to locate components that will first identify properties of the agent and the environment to monitor, before monitoring them over time to identify symptoms of emergence that may arise. Therefore,

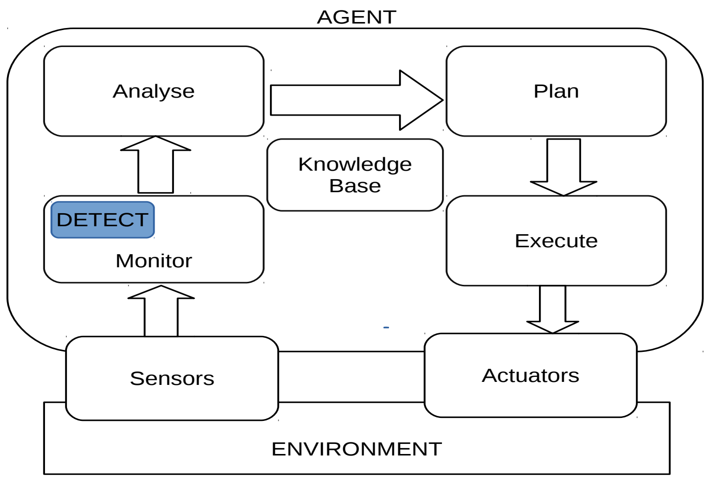

Fig. 3.2: ***DETect's position in a MAPE loop*** *executed by an agent [Kephart and Chess, 2003]*

DETect is located in this phase of the MAPE loop, with the output potentially triggering adaptations later in the adaptation work flow. This decision addresses design objective 2, allowing the detection of emergence to be useful in the system at run time. As a result of this decision, deciding what to do once emergence is detected is outside the scope of DETect.

3.3.3 Detecting Feedback From Emergence

3.3.3.1 Modelling Technique

Each agent has a limited view of the system, composed of its immediate locality and other agents nearby. It is in this window that the feedback from emergence through downward causation manifests, having the affect of constraining the agent [Bedau, 2002]. The hypothesis of this thesis is that this constraint will result in a changed statistical relationship between the agent and its environment, as experienced in its locality. Initial exploration of this concept [O'Toole et al., 2014], investigated whether specific statistical relationships, correlations, between the agent and its environment would form during periods of emergence. The agent was characterised by variables that described its internal state (Internal Variables) and the environment was described by a different set of variables (External Variables) This approach proved successful, with a higher proportion of agents experiencing such correlations during times of emergence compared to periods when no emergence was present in the system. However, its utility was limited by the need to specify in advance what variables to observe and the exact relationships between them to look for (statistically significant correlations). Furthermore, this initial approach relied on a centralised architecture as agents reported the detection of a significant correlation to a central observer and did not conclude themselves that emergence existed.

Instead, what is required is a statistical method that can generally describe the agent's relationship with its environment, allowing statistically significant changes in the relationship to be observed over time. In other words, a technique that does not look for a specific relationship but instead creates a baseline of what the relationship typically is and looks for deviations from this baseline. This task is complicated by the fact that some agents may use hundreds of variables to describe themselves and hundreds more to describe their environment, meaning that there are possibly thousands of relationships between these two variable sets. It would be computationally expensive for agents to monitor all of these individual relationships separately at runtime. Therefore, a simplified statistical model is required that allows multiple relationships to be represented at once, as well as a means of selecting only those relationships that provide relevant information to be included in the model.

Multiple-Linear Regression (MLR) [Draper and Smith, 1981], outlined in equation 3.1, provides a statistical model of the relationship between a response variable, y_i, and two or more explanatory variables, X={$\mathrm{x}_1 ... \mathrm{x}_n$}, by fitting a linear equation to the observed data. The coefficients β are the expected change in the response variable for every 1 unit change in the respective explanatory variable. This model describes how the mean of y_i responds when the explanatory variables are changed. Additionally, MLR outputs a p-value for each explanatory variable in the model. The p-value is used to evaluate the modelling hypothesis that the coefficient for the associated explanatory variable is zero, i.e. that there is no effect. It represents the probability of having observed the gathered data, or data even more extreme, if this hypothesis is true. Therefore, if the p-value is low (close to zero) the data appears inconsistent with the hypothesis of no-effect being present and we may wish to reject the hypothesis.

$$y_i = \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon_i \tag{3.1}$$

MLR represents an evolution from the initial exploration described in [O'Toole et al., 2014], with the correlations tests used there also quantifying the linear relationship between two variables. However, MLR allows multiple relationships between variables to be evaluated at once, while performing periodic regression tests, and observing the p-values produced over time, can indicate that the significance of the relationship between variables has changed. This analysis motivates the next design decision:

> **Design Decision 3 - MLR used as statistical model** Multiple-Linear Regression provides a general statistical model that can incorporate the variables that describe the agent and the variables that describe the agent's environment. This improves on correlation analysis that formed the basis of the early approach adopted in this research as it does not look for the appearance of a specific type of relationship between variables i.e., statistically significant correlations. In this way it reflects the challenge presented by emergence which is fundamentally unpredictable, so knowing exactly what kind of relationship to look for is not always possible. In addition, MLR allows multiple relationships to be evaluated at once instead of analysing relationships between an internal and all external variables one at a time, as would be the case with correlation analysis or simple linear models. As a result, it allows the relationship between the agent and the environment to be measured using only information locally available to the agent (design objective 3).

The decision to use MLR means that a trade off is made between simplicity and accuracy, as the model does not account for non-linear relationships that may exist between variables. This is discussed in more detail in Section 3.5 below.

3.3.3.2 Model Selection

Although MLR provides a simple way of monitoring the relationship between many variables, it does not provide a means of choosing what variables should be in the model. Some variables do not have any relationship to one another, for example, the direction of travel of a flocking agent and the agent's age. In this case, the agent could potentially ignore that relationship as it does not offer value. However, without any semantic understanding of this in advance, the agent needs to discover this at runtime.

The accuracy and validity of MLR can suffer if the explanatory variables, X, contain multi-collinearity. This means that linear relationships exist between individual explanatory variables, so changes to one affect the other. Multi-collinearity does not impact the reliability or predictive power of the overall model. However, it can affect the reliability of calculations of the effect each explanatory variable has on the response variable, which is the focus of concern here. As an example of this, a MLR model in a flocking agent can be imaged, where Y is the agent's speed and the explanatory variables contain variable *X1*, the number of other agents in the agent's field of vision and *X2*, the number of total agents in a defined radius of the agent. It is intuitive to think that an increase in *X1* will also likely mean an increase in *X2*, however the agent requires a means of discovering this relationship autonomously before taking appropriate action.

There are a number of possible model selection approaches, each with their own merits and limitations. A number of candidates are now presented and analysed in the context of this specific problem.

P-values generated by MLR provide a simple means of determining the significance of the relationship between the response variable Y and an explanatory variable X. When the p-value is low, close to zero, the linear relationship between the two variables is said to be more significant, where changes to X will likely result in changes in Y. Conversely, when the p-value is close to 1 no such relationship between the variables exists. Using this characteristic of p-values, a crude mechanism using a p-value threshold could be employed where relationships above a certain p-value are excluded. However, while such an approach would help to reduce the number of relationships an agent has to monitor, it would not address the issue of multi-collinearity between explanatory variables and some method of selecting an appropriate threshold would be required.

Information Theoretical Methods Akaike information criterion (AIC) [Akaike, 1973] or the Bayesian information criterion [Albert and Chib, 1997], are information theoretical and statistical methods of selecting between a finite set of possible models, by selecting the model with the lowest criterion. For example, in AIC an estimate of each model relative to all other models is provided using a trade off between how complex the model is and how well it fits the data. A drawback with these methods however is that all candidate models must be compared against each other. When an agent has many internal and external variables, this could require thousands of potential models to be compared which is likely to be inefficient both computationally and in terms of timeliness.

Principal Component Analysis (PCA) [Wold et al., 1987] is a statistical and machine learning technique that can be used for dimensionality reduction or feature selection. As its name suggests, PCA finds the principal components of the data by linearly transforming a d-dimensional dataset into a k-dimensional subspace, where $k < d$. In doing so, the variables in the dataset are replaced with "principal components", which are linear combinations of the original variables that retain as much information as possible. In doing so, the computational efficiency of subsequent analysis on that dataset is improved while most of the information in the dataset is retained. If the intention is to use PCA for feature selection, as is the intended purpose here, future datasets should be transformed in the same manner as the initial dataset so as to create same principal components. In context of periodic analysis of the model, this would increase the computational cost of each subsequent analysis step after the initial PCA has been executed.

Least Absolute Shrinkage and Selection Operator (LASSO) is a shrinkage and selection method for use with linear regression [Tibshirani, 1996] to prevent overfitting. LASSO, outlined in equation 3.2, returns the set of coefficients, β^{lasso}, that minimises the sum of squared errors while a regularisation penalty, λ, is applied to the absolute value of each of the coefficients, β. This penalty results in all coefficients being shrunk toward zero and some coefficients will be exactly zero meaning they have no affect on the model and can be discarded. As a result, LASSO provides automatic variable selection while also dealing with multicollinearity in the explanatory variables by removing variables that do not add information to the model.

$$\beta^{lasso} = arg \min_{\beta} \sum_{i=1}^{n} (y_i - (\beta^T xi))^2 + \lambda \|\beta\|_1 \tag{3.2}$$

Design Decision 4 **- LASSO is used to select the model** From analysis of these techniques and in the context of this thesis, LASSO provides the best trade-off between accuracy and efficiency of the feature selection techniques discussed. LASSO is designed to work with linear regression and automatically provides variable selection while also handling multi-collinearity in the data. Additionally, once variables have been selected, future analysis of the model will not require the dataset to be transformed. This feature [provides a significant advantage compared to PCA which would require the data to be transformed into the principle components each time the model is evaluated throughout the systems evolution, increasing the computational cost to each agent. LASSO also does not require all potential models to be compared against one another in contrast to Information Theoretical Methods. This feature means it is more suited in this specific application where model selection occurs at run time and there is no limit to the number of theoretical candidate variables that could compose the final model. This design decision addresses design objective 2 by allowing variables to be selected at run time, and design objective 5 by providing a method of autonomous variable selection.

3.3.3.3 Feedback Detection

Due to the nature of emergence, it is not possible to predict in advance what form or shape it will take in any given system. This means that agents should be limited to only detecting changes in the emergent state of the system, for example when emergence forms and when emergence evaporates. To clarify this point, consider the example of flocking agents. Using the modelling approach discussed above, the agents could model their relationship with the environment when there is no flocking. They could also do this when flocking is forming, when flocking is present or when flocking is evaporating. However, without a semantic description of the emergent system behaviour in advance, i.e. what it means to "flock" and how it will impact them, each agent has no way of attributing each model of the system to a particular emergent system state. As a result, agents should not try to detect the presence of emergent behaviour, but rather changes in the emergent behaviour that are reflected in a changed relationship with their environment.

Once an agent has modelled its relationship to the environment, it is necessary to monitor this relationship over time to detect such changes. Achieving this requires that the agent should create a range of typical or expected measurements for the relationship against which periodic evaluations can be compared. This allows for significant departures from the expected range to be quickly diagnosed. This range of expected values must be constantly updated to reflect the transient nature of emergence. For example, flocking behaviour in birds both forms and evaporates quickly. Detecting when both of these events occur is necessary to ensure that the baseline of expected values is accurate. For example, before flocking forms, the characteristics of the system without flocking behaviour should provide the standard against which an agent judges their current relationship. Once flocking has occurred, the agent should use the characteristics of the system with flocking to compare against, enabling it to detect when the emergent behaviour evaporates.

This means the agent is required to define what constitutes a significant change from this standard. This can be achieved using a number of existing change point detection algorithms [Kawahara et al., 2007, Adams and MacKay, 2007], which identify when the probability distribution of a time series of values changes. Cumulative Sum (CUSUM) [Page, 1954], is a computationally inexpensive on-line change detection method that involves sequentially summing time series samples, x, as outlined in equation 3.3. S_0 is initialised to 0 and K refers to a weighting that is applied to each sample. When S exceeds a defined threshold, h, a change point in the series has been found. CUSUM can be used to identify significant departures from the expected mean or standard deviation of a process and is frequently used in quality control procedures [Healy, 1987]. Non-parametrised versions of CUSUM are also possible, meaning that the expected mean and standard deviation of the process does not have to be specified in advance [Hawkins, 1987]. Therefore, with regard to feedback detection, the next two design decisions are as follows:

$$S_{n+1} = max(0, S_n + x_n - K) \tag{3.3}$$

Design Decision 5 **- A sliding window approach to monitoring** Figure 3.3 illustrates a sliding window approach to monitoring. A sliding observation window of length N is maintained and updated every time step until full. When full, the M oldest observations are deleted from the window allowing the window to "slide" gradually as the system evolves. This monitoring approach is similar to that described by Grossman et al. [Grossman et al., 2009]. In the context of this thesis, the sliding observation window would constitute the baseline against which each agent judges its current experience. This allows for the history of the system to be taken into account without requiring the entire history to be maintained. Additionally, gradual updates to the baseline can in principal allow transitions in the emergent state to be detected, addressing design objective 6. The choice of values for both N and M are discussed later in this chapter.

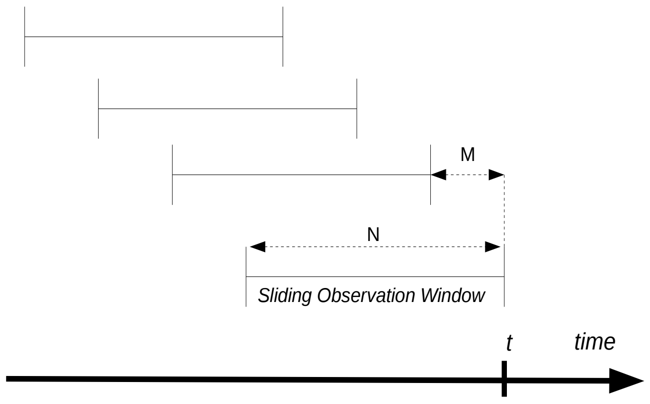

Fig. 3.3: ***Sliding window approach:*** *A sliding window of length N with updates of length M.*

Design Decision 6 **- CUSUM used to detect change** In conjunction with design decision 5, CUSUM can be used to judge current observations of the agent's relationship with its environment against the recent history of the relationship, as contained in the sliding observation window. As CUSUM is an on-line technique, it allows detection to occur at runtime, addressing design objective 2. In addition, it can be used to detect the transition between periods of emergence and periods of no emergence in the system, addressing design objective 6.

3.3.4 Consensus Formation

When an agent detects a change in its relationship with the environment, this indicates that some emergent property or behaviour *may* be present in the system. However, an individual agent is unable to truly reason about the existence of emergence due to the limited scope of their view of the system, while emergent property and behaviour occur at the global system level. This global aspect of the emergence means that it will impact more than one agent simultaneously when it occurs. This provides agents with a means of checking their local experience in the context of a wider view that is presented when multiple agents aggregate their individual system views together. An agent should only suspect emergence if there is a consensus amongst its peers that an emergent may be present in the system. For example, if a single car has stopped for a prolonged period of time this does not necessarily mean a traffic-jam exists.

To achieve this requires a decentralised communication protocol that allows agents to identify other agents, and share information about their changing relationship with their environment. As the existence of emergence is often transient in nature e.g. flocking behaviour can form and evaporate quickly, the time taken in achieving consensus is of critical importance. This is compounded by the possibility that each agent may detect feedback at slightly different times depending on their individual experience, meaning that some resilience must be built into the protocol to handle this. Finally, the dynamic nature of CAS with an ever changing topology means that deciding who to exchange information with, must also be considered.

Three alternative communication protocols are identified that are now evaluated in the context of the specific requirements of emergence and CAS. The protocols under consideration are:

Gradient fields are an information propagation and coordination mechanism that allows information about a senders to be propagated, typically by incrementing or decrementing some property to reflect distance or concentration. They have been used in domains such as coordination of robots [Parunak et al., 2002] or routing in networks [Mamei and Zambonelli, 2004]. In addition, they can be used to implement quorum sensing [Fernandez-Marquez et al., 2012], a biologically inspired decentralised decision making process for coordinating behaviour and making collective decisions [Miller and Bassler, 2001, Seeley and Visscher, 2004]. Its goal is to provide an estimation of the number or density of agents in a system using only local interactions. This could be potentially useful in the context of this thesis, providing agents with a way of estimating the number of other agents who have recently detected feedback from emergence.

Digital pheromone is a coordination mechanism based on indirect communication that takes its inspiration from ant colonies [Grassé, 1959, Dorigo et al., 2000]. Pheromones are deposited by agents into the environment that spread a gradient over the environment and persist for a finite period of time before fading away. They have been used in a number of decentralised systems in domains such as scheduling [Martens et al., 2007], vehicle routing [Toth and Vigo, 2002] and control [Sauter et al., 2005]. In the context of this thesis, pheromones provide a potential means of allowing agents to communicate the fact that they have detected a change without requiring direct communication with other agents in the network. Instead, agents could potentially sense this information from the environment with the intensity of local pheromones indirectly informing agents of the number of agents who have recently experienced a similar change.

Gossip-based consensus protocols have widely been used across a variety of networks types, proving efficient in both large scale networks [Jelasity et al., 2005, Nedos et al., 2006] and mobile ad-hoc networks [Datta et al., 2004]. The fundamental component of the protocols involves periodic, pairwise, interactions between agents with information exchanged during these interactions being of bounded size. When agents interact, the state of at least one of the agents changes to incorporate the information received from the other agent. In this way the knowledge of individual agents increases while uncertainty decreases, allowing agreement to be reached about the value of some system properties in a decentralised way. Gossip-based protocols can be used to calculate aggregate information across networks such as summations and averages [Kempe et al., 2003] and are resilient to node failures. Gossip-based protocols have also been used to facilitate consensus formation in networks without the need for a centralised controller [Carli et al., 2010, Lavaei and Murray, 2012].

Based on these options, the final design decision is as follows:

> **Design Decision 7 - Gossip based communication** Gossip based consensus is a widely used communication protocol in dynamic networks such as the systems that are the focus of this thesis. In particular, it does not require additional effort to specify how gradients should be propagated across the system and environment. Its biggest advantage over both gradient fields and digital pheromones is that it does not require any additional infrastructure or special agents to be introduced. For example, digital pheromones require some form of shared platform in which the pheromones can be placed. This could be achieved using a middleware or tuple-space [Gelernter and Carriero, 1992], however this introduces additional assumptions and complexity which is not required with gossip. The use of gossip-based communication addresses design objective 1 and 4.

Figure 3.4 summarises the design decisions that constitute the novel distributed algorithm presented in this thesis and maps them to their underlying design objective outlined earlier in the chapter (*cf.,* Section 3.1). In the next section, the proposed solution derived from these design decisions is described.

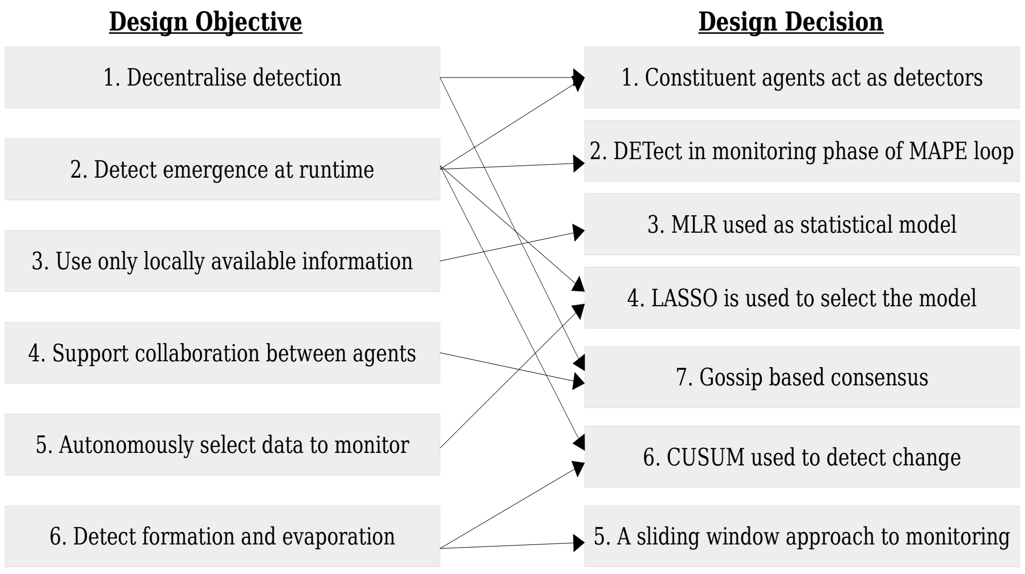
Design Objective
Design Decision
1. Decentralise detection
2. Detect emergence at runtime
3. Use only locally available information
4. Support collaboration between agents
5. Autonomously select data to monitor
6. Detect formation and evaporation
1. Constituent agents act as detectors
2. DETect in monitoring phase of MAPE loop
3. MLR used as statistical model
4. LASSO is used to select the model
7. Gossip based consensus
6. CUSUM used to detect change
5. A sliding window approach to monitoring

Fig. 3.4: ***Design Decisions*** *mapped to motivating Design Objectives.*

3.4 Proposed Solution

This thesis presents Decentralised Emergence Detection (DETect), a novel distributed algorithm that enables agents to collaboratively detect emergent events in CAS. Emergent event refers to either the formation or evaporation of emergence in the system. The foundation of this detection technique is built on enabling individual agents to detect feedback from emergence, from downward causation, in the local environment. DETect employs a modular design, illustrated in Figure 3.5, that is intended to deliver three key competences. The *Modelling Unit* provides agents with a general method of modelling their relationship with their local environment, autonomously selecting what properties of both the agent and the environment to monitor at run time. Next, the *Change Detection Unit*, provides agents with an on-line means of detecting a significant change in this selected relationship over time. Finally, the *Collaboration Unit* provides a decentralised gossip-based communication mechanism that allows agents to share information about detected feedback and thus build consensus on the existence of emergence, without the need for a central observer. The remainder of this section describes each of these components in detail.

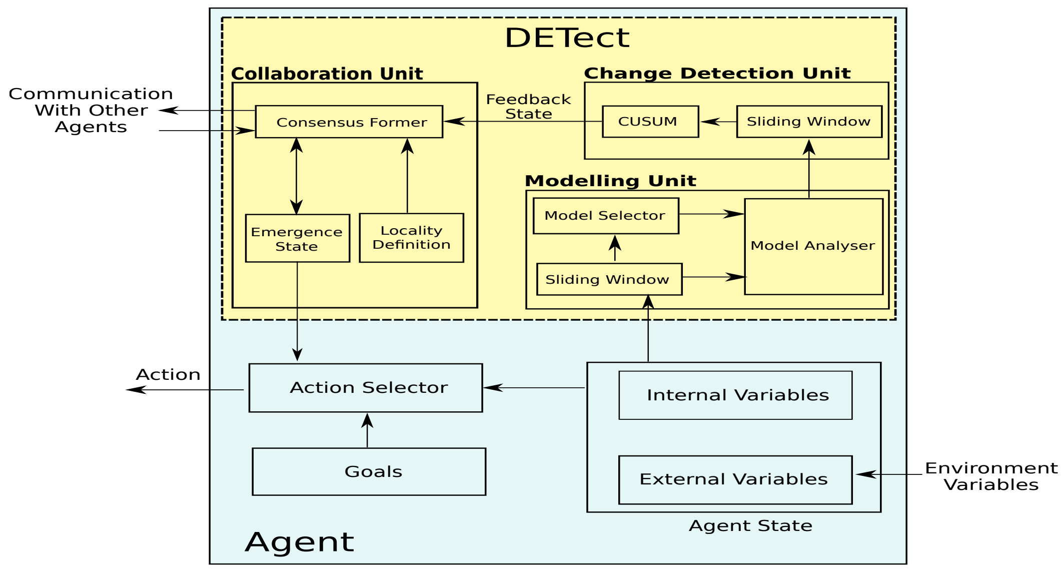

Fig. 3.5: ***DETect - Conceptual Architecture***

3.4.1 Modelling Unit

The *Modelling Unit* performs three functions as part of its role in DETect. First, it is DETect's primary interface with the agent, receiving both internal and external variable sets and initialising variable observation. Next, it selects the model that will be used to characterise the agent's relationship with its environment. Finally, it periodically analyses this model using MLR, passing the results onto the *Change Detection Unit*. The flow of information through the Modelling Unit is illustrated in Figure 3.6, demonstrating how each of these three functions integrate with one another.

3.4.1.1 Initialisation and Variable Observation

The initialisation phase and variable observation process in DETect is handled using a *Sliding Window*. This approach uses a sliding observation window of length N that is updated when full by deleting the M oldest values in the window. DETect uses a number of sliding windows throughout its work flow with the values of M and N changing depending on the window being used. This is discussed in detail in the sections below.

The initialisation and variable observation process on each agent is detailed in Algorithm 1.

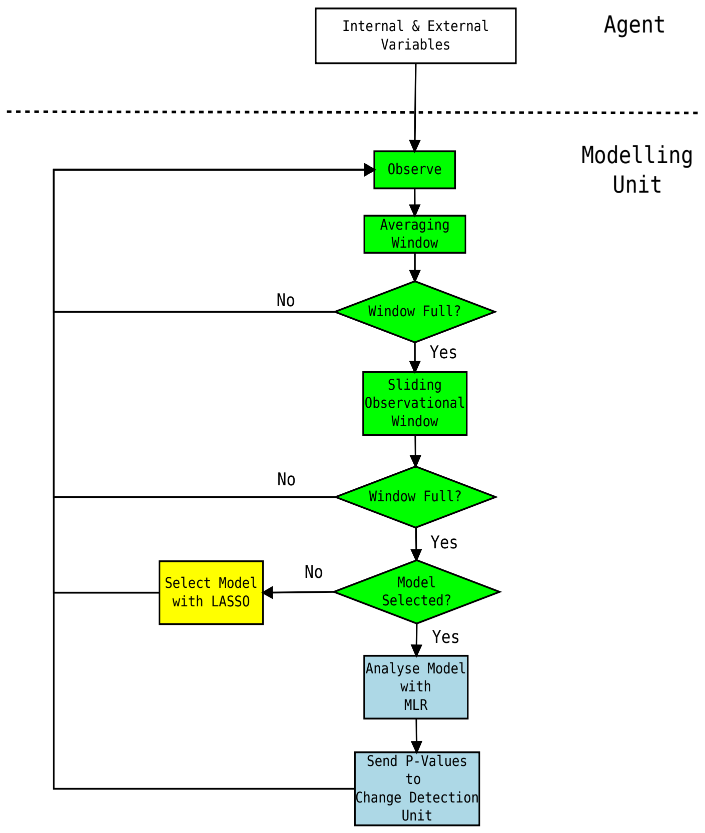

Fig. 3.6: ***Flow of information in DETect's Modelling Unit:*** *Algorithm 1 is highlighted in green, Algorithm 2 in yellow and Algorithm 3 in blue.*

To improve readability, all Internal Variables and External Variables of the agent are grouped into a single set of variables, as the process applied to both sets of variables is the same in this algorithm. It is assumed that this set of variables does not change during the system evolution. The impact of this assumption is discussed later in the chapter when the limitations associated with this design are outlined (*cf.,* Section 3.5).

DETect begins by initialising a pair of arrays for all variables used by the agent (lines 2-5). The first of these, the *AvgArray*, is used to average a group of consecutive observations for each variable in order to improve the normality of data. This averaging is done as a preprocessing step for MLR, as normality is an important assumption made when using any linear regression technique. The second array is the sliding observation window $SlidingObsWindow$. The max-

Algorithm 1 DETect Initialisation and Variable Monitoring on Agent A_i

```
Input: All variables $V_{i1}$ to $V_{iv}$ of agent $A_i$. Averaging window size $AWS$. Sliding observation window size $N$. How
    far the Sliding Window should move once full $M$.
Output: The sliding observation window of variable $V_i$ is filled
 1: $ModelSelected = FALSE$ // Boolean value indicating if the regression model has been selected
 2: for all $V_{i1}$ to $V_{iv}$ do // Initialise Windows for all Variables
 3:     Initialise empty $AvgArray_{iv}$;
 4:     Initialise empty $SlidingObsWindow_{iv}$;
 5: end for
 6: for all time steps 1 to t do // Observe variables at each time step
 7:     for $V_{i1}$ to $V_{ip}$ do
 8:         Add Observation$_{ipt}$ to $AvgArray_{ip}$;
 9:     end for
10:     if $size(AvgArray_{i1}) == AWS$ then // All AvgArrays are the same size
11:         for $V_{i1}$ to $V_{iv}$ do
12:             /* Average Observations and Add to Sliding Observation Window /*
13:             Add $mean(AvgArray_{iv})$ to $SlidingObsWindow_{iv}$;
14:             Initialise empty $AvgArray_{iv}$
15:         end for
16:         if $size(SlidingObsWindow_{i1}) == N$ then
17:             /* Send data for analysis if model has been selected */
18:             if $ModelSelected$ then
19:                 analyseModel($SlidingObsWindow_{i1}$ to $SlidingObsWindow_{iv}$)
20:             else
21:                 $ModelSelected$ = selectModel($SlidingObsWindow_{i1}$ to $SlidingObsWindow_{iv}$)
22:             end if
23:             /* Slide observation windows */
24:             for $V_{i1}$ to $V_{iv}$ do
25:                 Delete $SlidingObsWindow_{iv}[1 : M]$;
26:             end for
27:         end if
28:     end if
29: end for
```

imum length of the *AvgArray* and *SlidingObsWindow* are specified by the parameters AWS and N respectively, meaning that when they reach this length they are considered to be "full".

A new observation for each variable is received at each time step of the system and is added to the relevant *AvgArray* (line 7-9). The *AvgArrays* for all variables have a common size as they are updated synchronously. Once the *AvgArrays* is full (line 10), the mean value is calculated and added to the respective *SlidingObsWindow* of each variable and the *AvgArray* is cleared (lines 11-15). Next, the *SlidingObsWindow* are checked to determine if they are full (line 16). If so, the windows for all variables are forwarded to either the *analyseModel* (*cf.,* Algorithm 3) or *selectModel* (*cf.,* Algorithm 2) functions. The *selectModel* algorithm will return a boolean value indicating whether the model has been selected successfully (line 21). Once this is done, the sliding windows are moved forward by deleting the earliest values in the window, corresponding to the size of the jump the window should make, indicated by parameter M (lines 24-26).

3.4.1.2 Model Selection

The next step in the DETect algorithm is to select a model of the agent's relationship with its environment that will be monitored over time for change detection. This occurs in the *Model Selector* component of the *Modelling Unit*, and is detailed in Algorithm 2. This process is considered the end of the initialisation stage of DETect and occurs when the *sliding observation window* reaches a size of $N = 500$, which corresponds to 2500 time steps. This window size was selected after preliminary experimentation to balance the goals of minimizing the length of the initialization phase and providing the *Modelling Unit* with sufficient information on which to build the model. Its size is consistent across all models described in this thesis and it is discussed in more detail in Chapter 5 (*cf.,* Section 5.1.1).

The model selection process is based on LASSO (*cf.,* Section 3.3.3.2). It begins by initialising a *model map* (line 1), that will be filled with internal-external variable pairs, representing the variable relationships that will form the regression model. Once this is complete, some pre-processing steps are applied to the data, which are intended to remove variables that are static (contain all the same values) or are comprised of just random noise. First, some zero-mean noise is added to each individual observation (line 2) to ensure that no entirely static variable is fed into the LASSO. Next, each individual internal and external variable is checked to see if it is just random noise (line 3). This is done by converting the observations for each variable to a distribution and using Pearson's Chi-square test for randomness [Pearson, 1900]. If a variable's variation is deemed to be entirely random, it is excluded from the model at this point as it does not contain any information. Following this, the number of internal and external variables is updated (lines 4-5) before the final pre-processing step centres and scales each variable's data, by subtracting its mean from each observation and dividing by its standard deviation (line 6).

Next, the remaining variables are fed to LASSO to choose the model. The LASSO is run in two stages, first for each internal variable against all external variables (lines 8-18), and then the process is repeated for each external variable against all internal variables (lines 19-27). The order of

Algorithm 2 Model Selection in DETect

```
Input: Internal Variables Sliding Windows SWI$_i$ and External Variables Sliding Windows SWE$_i$ of agent A$_i$. The
    number of external variables $L$. The number of internal variables $K$.
Output: $modelMap$ containing all internal-external variable pairs used in the agents model
 1: modelMap = new Map(InternalVariable,ExternalVariable) // Initialise Model Chosen Map
 2: injectNoise(SWI$_i$, SWE$_i$) // Inject Noise
 3: removeRandomVariables(SWI$_i$, SWE$_i$) // Remove variables containing only noise
 4: $L$ = count(SWE$_i$) // Update Variable Counts
 5: $K$ = count(SWI$_i$)
 6: centerAndScale (SWI$_i$, SWE$_i$) // Subtract the mean from each variable and divide by standard deviation
 7: $LassoCoefficientsMatrix$ = 2DArray[$K$ * $L$]
 8: for $SWI_{i1}$ to $SWI_{iK}$ do // Run LASSO For each Internal Variable
 9:     $Ym = SWI_{il}$;
10:     Initialise $LassoCoefficient$ = new Array[L];
11:     $LassoCoefficient$ = RunLasso($Yk$,SWE$_i$);
12:     for $LassoCoefficient[l]$ to $LassoCoefficient[L]$ do
13:         /* Check Coefficient To See What Models Were Selected */
14:         if LassoCoefficient[l] != 0 then
15:             $LassoCoefficientsMatrix[i,l] = LassoCoefficientsMatrix[i,l] + 1$
16:         end if
17:     end for
18: end for
19: for $SWE_{i1}$ to $SWE_{iL}$ do // Run LASSO For each External Variable
20:     $Yl = SWE_{il}$;
21:     $LassoCoefficient$ = RunLasso($Yl$,SWI$_i$);
22:     for $LassoCoefficient[1]$ to $LassoCoefficient[K]$ do
23:         if LassoCoefficient[k] != 0 then
24:             $LassoCoefficientsMatrix[k,l] = LassoCoefficientsMatrix[k,l] + 1$
25:         end if
26:     end for
27: end for
28: for each cell in $LassoCoefficientsMatrix$ do // Update the Model Map
29:     if cell == 2 then
30:         Add InternalVariable[Row-Number], ExternalVariable [ColumnNumber] to $modelMap$
31:     end if
32: end for
33: if size($modelMap$) > 0 then
34:     return($TRUE$)
35: else
36:     return($FALSE$)
37: end if
```

these two stages is irrelevant to the outcome of the model selection process i.e., external variables against all internal variables could be evaluated first. A matrix, $LassoCoefficientsMatrix$, is initialised (line 7) with zeros, where each row corresponds to an internal variable and each column corresponds to its relationship with an external variable. In the first stage, the observations for each internal variable is taken (line 9) as well as the observations for all external variables. Next an array is initialised that will be used to store the output of LASSO (line 10). LASSO works by returning the coefficients, β, for all external variables as fitted by the LASSO algorithm (line 11). Next, the coefficients are checked in turn and if a coefficient is non-zero, the corresponding entry in $LassoCoefficientsMatrix$ is updated by adding 1 (lines 14-16). The regularisation term, λ, is selected independently for each Lasso run using cross-validation, meaning it does not have to be set in advance.

In the second stage, this process is repeated for each external variable against all internal variables, with the $LassoCoefficientsMatrix$ once again updated by 1 should LASSO return a non-zero coefficient for the corresponding internal-external variable pair (lines 23-25). Finally the coefficients matrix is checked to see what cells contain a value of 2, indicating that LASSO returned a non-zero coefficient for the internal-external variable pair for both LASSO runs (lines 28-32). Variable pairs that have these relationships are added to the model map (line 30). Finally, if the model contains at least 1 internal-external variable pair the model selection is deemed successful and a boolean $True$ is returned (line 33-37).

As a result an internal variable can be added to the modelled variables set and a list of all external variables relevant to that variable is maintained. It may be that no relationships are selected when this algorithm is run. If that is the case, model selection is re-run every time the variable sliding windows are filled until at least one internal-external variable relationship is selected. The design assumes that eventually at least one variable pair will be selected by this process (*cf.,* Section 3.5). As a result of this process, an agent's relationship model with its environment in DETect is represented as one or more MLR models, each of which is independently monitored for change as the system runs.

3.4.1.3 Model Analysis

After the model of the relationship between the agent and its environment has been selected, the sliding observation window becomes the $RegressionWindows$ and is reduced to the size of this parameter. Various sizes of the $RegressionWindows$ are tested as part of the case study described in Chapter 5. All subsequent instances of the $Regressionwindows$ being filled results in the selected model being analysed using MLR. This process is outlined in Algorithm 3, with

Algorithm 3 Analyse Model

```
Input: $modelMap$ containing all internal-external variable pairs used in the agents model. Internal Variables Regression
    Window SWI$_i$ and External Variables Regression Window SWE$_i$ of agent $A_i$.
Output: P-value time-series, $newP_{ijs}$ for all internal-external variable pairs contained in $modelMap$
 1: /* Initialise */
 2: $modelInternalVars$ = Internal Variables in $modelMap$;
 3: $newP_{ijs}$ = empty matrix array;
 4: /* Perform MLR */
 5: for each $modelInternalVars$ to All do
 6:     Y = $SWI_{ij}$ of $modelInternalVars_j$;
 7:     X = empty array;
 8:     for all $E_{is}$ of $modelInternalVars_j$ in $modelMap$ do
 9:         X = X + $SWE_{is}$;
10:     end for
11:     $newP_{ijs}$ = MLR(Y,X);
12: end for
```

the *Model Analyser* component taking the set of interval variables chosen by the *Model Selector* and fitting a MLR model for each internal variable chosen in Algorithm 2. For each MLR Y is the internal variable, and the explanatory variables X are the external variables associated with it in the *modelMap* (lines 5-12). The output of this MLR is a set of p-values for each internal-external variable pair, with these p-values representing the significance of the relationship between the variables at that time (line 11). These p-values are forwarded to the *Change Detector Unit*, where they are monitored over time for change to determine if feedback is present in the environment. By collecting p-values over time, a model of the significance of a relationship can be formed, making it possible to compare against future measurements. For example, if the historical p-values for a relationship are consistently high, a sudden drop in the p-values in recent analysis will indicate that something may have happened to change the significance level of the relationship.

3.4.2 Change Detection Unit

The *Change Detection Unit* is responsible for monitoring the p-values associated with each internal-external variable pair and detecting when a deviation from an expected baseline has occurred. Any significant change indicates that the agent's relationship with its environment may have changed resulting in a *feedback detection* event being generated and sent to the *Collaboration Unit* (*cf.,* Section 3.4.3). Change detection is achieved using a combination of a Cumulative Sum (CUSUM) analysis and a sliding window for the p-values generated by each internal-external variable pair, the *CUSUM Windows*.

Hawkins describes a self-starting on-line CUSUM algorithm [Hawkins, 1987], that calculates the CUSUM for a process variable as each new observation of the variable arrives, to determine if the most recent observations of the process indicate a deviation from its expected mean or standard deviation. These expected values are the mean and standard deviation of all historic values of the variable generated by the process up to that point. Once the CUSUM is calculated, the expected mean and standard deviation is updated to include the new observation and these updated values act as the baseline for the next time an observation arrives and the CUSUM is recalculated.

Hawkins' approach provides the foundation of the CUSUM method used by the *Change Detection Unit* in DETect. This approach is applied separately to the p-values of each internal-external variable pair relationship contained in the DETect model, to determine when the mean or standard deviation of the p-values changes significantly. However, **unlike Hawkins' approach**, the *Change Detection Unit* does not use the mean and standard deviations of all historic observations of the p-value and instead uses the N most recent, the *CUSUM Window*, to calculate the expected mean and standard deviation. The reason for not basing the mean and standard deviation on all historic observations of the p-value is that doing so would ultimately result in the expected baseline becoming too broad as the system evolves.

Taking flocking as an example, the expected mean and standard deviation would eventually incorporate p-values from periods with flocking, without flocking, and transition periods, and therefore detecting the transition between these periods would become impossible using the CUSUM. By using only recent historic values, the *CUSUM Window*, this problem is avoided assuming that the *CUSUM Window* is sufficiently narrow so that it does not overlap these different macro-level states. However, the *CUSUM Window* should also be sufficiently wide so as to represent a reasonable baseline and to ensure that the CUSUM analysis is not too sensitive, which would result in a change being suspected very frequently. The size of the *CUSUM Window*, N, is therefore a critical parameter in DETect and it is one of the parameters evaluated in the case study described in Chapter 5, with the effect of a range of different *CUSUM Window* sizes analysed (*cf.,* Section 5.1.3).

The algorithm used by the *Change Detection Unit* is described in Algorithm 4. It takes as input the model selected by DETect containing the names of all internal variables and their associated external variables. It also takes some parameters, including the specific value of N to be used for the *CUSUM Window* and the parameters K and h used by the CUSUM algorithm. Setting the values for both the weighting parameter, K and the CUSUM threshold, h, is important to the performance and sensitivity of the CUSUM (*cf.,* Section 3.3.3.3). It is typical to set a value of K to half the number of standard deviations from the mean. Assuming a normal distribution in the data, 95% of values can be expected to fall within 2 standard deviations of the mean. Using this 5% significance level as a threshold, $K = 1$ was selected, and used throughout the evaluation of DETect. The threshold was set to $h = 4$ after preliminary experimentation involving the Traffic model described in Section 4.5 to find a reasonable level for the parameter.

The first section of Algorithm 4 (lines 1-11) sets up the *CUSUM Window* and cumulative sums, *SMean* and *Sdev* for each internal-external variable pair contained in the *model map*. *SMean* (line 8) is used to look for deviations from the expected mean of the p-value and *SDev* (line 9) is used to look for deviations from the expected standard deviation of the p-value as the system evolves. The main CUSUM section begins at line 12. A check is made at every time step to determine if a new set of p-values have been generated for the model by the modelling unit (line 14). If this is the case, the feedback detection event flag is set to false before the analysis takes place (line 15). Next, each *CUSUM Window* being monitored is processed in turn, with the specific p-value associated with that variable pair being extracted from the received set of p-values (line 17). If the *CUSUM Window* is full, the expected mean and standard deviation are calculated and the CUSUM analysis is executed (lines 19-21) The CUSUM analysis returns the updated CUSUM values for both *SMean* and *SDev* and the absolute value of these is compared to the CUSUM threshold h, to determine if a significant change has occurred (line 22-24). Next, the oldest value in the *CUSUM Window* is deleted (line 25), before the most recent observation

Algorithm 4 Relationship Monitoring and Change Detection

Input: The model being observed, $modelMap$, containing the names of the internal-external variable pairs. New p-values for each internal-external variable pair in the $newP_{ijs}$ in $modelMap$. The size of the CUSUM sliding window N_{cusum}. h the CUSUM threshold. K CUSUM weighting parameter

Output: A feedback detection event, $feedBackFound$

```
 1: /* Initialise */
 2: $modelInternalVars$ = Internal Variables in $modelMap$;
 3: for all $modelInternalVars$ $MIV_a$ do
 4:     $modelEnternalVars = modelMap$.get($MIV_a$); // All external variables associated with internal variable
 5:     for all $modelEnternalVars$ $MEV_b$ do
 6:         initialise empty $CusumWindow_{ab}$
 7:         /* Initialise CUSUM for the mean and standard deviation */
 8:         $Smean_{ab} = 0$
 9:         $SDev_{ab} = 0$
10:     end for
11: end for
12: /* Main CUSUM Part */
13: for all Time steps do
14:     if $newP_{ijs}$ arrived then
15:         $feedBackFound$ = false
16:         for all $CusumWindow_{ab}$ do
17:             $newP_{ab}$ = get P-value from $newP_{ijs}$
18:             if size($CusumWindow_{ab}$) == $N_{cusum}$ then
19:                 expectedMean = mean($CusumWindow_{ab}$)
20:                 expectedSd = sd($CusumWindow_{ab}$)
21:                 $[SMean_{ab}, SDev_{ab}]$ = CUSUM(expectedMean,expectedSd,$SMean_{ab}$,$SDev_{ab}$,$K$)
22:                 if absolute($SMean_{ab}$) > $h$ or absolute($SMean_{ab}$) > $h$ then
23:                     $feedBackFound$ = true
24:                 end if
25:                 Delete $CusumWindow_{ab}[0]$
26:             end if
27:             $CusumWindow_{ab}$.add($newP_{ab}$) // Add most recent p-value to the window
28:         end for
29:         send $feedBackFound$ to Collaboration Unit
30:     end if
31: end for
```

of the p-value is added to the end of the window (line 27). Finally, the state of *feedBackFound* is sent to the *Collaboration Unit*, which indicates if a statistical change has occurred in the agents relationship with its environment (line 29).

3.4.3 Collaboration Unit

The final component of DETect is the *Collaboration Unit*, which is responsible for inter-agent communication in order to share information about feedback detection and thus allow agents to learn about the state of the system beyond their immediate locality. Algorithm 5 outlines a gossiping consensus algorithm, based on distributed averaging, that allows agents to reach consensus on the presence of sufficient feedback detection in the system. When DETect raises a feedback detection event, the variable Lv is set to 1 to reflect this (lines $3-7$). DETect also maintains a second variable, Ev, which represents the proportion of other agents in the system believed to have also detected feedback. Ev is updated by randomly selecting one of the agent's neighbours and updating both Ev values to their average (lines $11-15$). After this is done, Ev is scaled towards the agent's own feedback variable Lv, meaning that without reinforcement from other agents, DETect will believe its own experience (lines $22-24$). DETect concludes the existence of an emergent event when Ev exceeds a certain threshold (lines $29-31$). The value of this threshold is explored as part of the evaluation described in Chapter 5 (*cf.,* Section 5.1.4).

DETect makes a number of compromises between speed and accuracy in consensus formation between agents (lines 9-15). To begin, an upper and lower limit is used on the neighbourhood size of the agent, meaning that DETect does not gossip unless the agent has a minimum number of neighbour agents. The is due to the nature of p-values which means that some random changes may occur in the monitored relationships and may cause individual agents to falsely detect change. If an agent is in a sparsely populated area of the environment it is possible that a disconnected sub-network may form due to limitation in communication range. With few one-hop neighbours, the consensus averaging for this group will be artificially high compared to a densely populated area if a random p-value change occurs. However, agent interactions are a fundamental requirement for emergence to exist [De Wolf and Holvoet, 2005], so by not trying to detect emergence when an agent has few neighbours and thus, few interactions, false-positive emergence detections such as these can be reduced. For example, such a group composed of only two agents would form a consensus of 50% of agents are experiencing a change if one of them experienced a random p-value change. Meanwhile, a group elsewhere in the system composed of 500 agents would form a consensus of only 40% if 200 of them simultaneously experienced a change. Finally, DETect remembers that feedback was detected for a period of time, allowing multiple agents who do not detect feedback at exactly the same time to still form consensus over

Algorithm 5 DETect Distributed Consensus Forming Step

Input: A feedback detection event, $feedBackFound$. The set N_{it} containing the one-hop neighbours of agent A_i at time t. DETect's previous estimate of feedback consensus in the system Ev. The minimum and maximum neighbourhood sizes permitted, $neighMin$ and $neighMax$. The consensus threshold value that triggers an emergence detection event $consensusTheshold$. How quickly to scale consensus belief to feedback value, $scaler$

Output: An emergence detection event $EmergeDetected$

```
1: /* Initialise */
2: $EmergeDetected = FALSE$
3: if $feedBackFound == TRUE$ then
4:     $Lv = 1$;
5: else
6:     $Lv = 0$;
7: end if
8: /* Check if there are sufficient neighbour agents */
9: if size($N_{it}$) $\geq neighMin$) then
10:     /* If there are too many neighbours, select the $neighMax$ closest */
11:     if size($N_{it}$) $\leq neighMax$) then
12:         $gossipCanidates = N_{it}$;
13:     else
14:         $gossipCanidates$ = nearest($neighMax$, $N_{it}$);
15:     end if
16:     /* Gossip */
17:     $partner = Random(gossipCanidates)$;
18:     $partnerEv = Ev$ of $partner$;
19:     $myEv = (partnerEv + Ev) / 2$;
20:     Ask $partner$ update $Ev$ to $myEv$;
21:     $Ev = myEv$;
22:     /* Scale to Own Feedback Detection Belief */
23:     $diff = Ev - Lv$
24:     $Ev = Ev - diff * scaler)$
25: else
26:     $Ev = 0$;
27: end if
28: /* Raise Emergence Detected Event if threshold exceeded */
29: if $Ev \geq consensusTheshold$ then
30:     $EmergeDetected = TRUE$;
31: end if
```

time.

DETect will execute Algorithm 5 at each time step that the agent has a sufficient number of neighbours. A single agent may act as a partner to more than one neighbour at each time step as the selection of a partner is random. As a consequence, the order in which interactions occur impacts the final value that an agent's value Ev will have at the end of each time step. Figure 3.7 illustrates the gossip work flow of interaction between 3 agents, where Agent 1 randomly selects

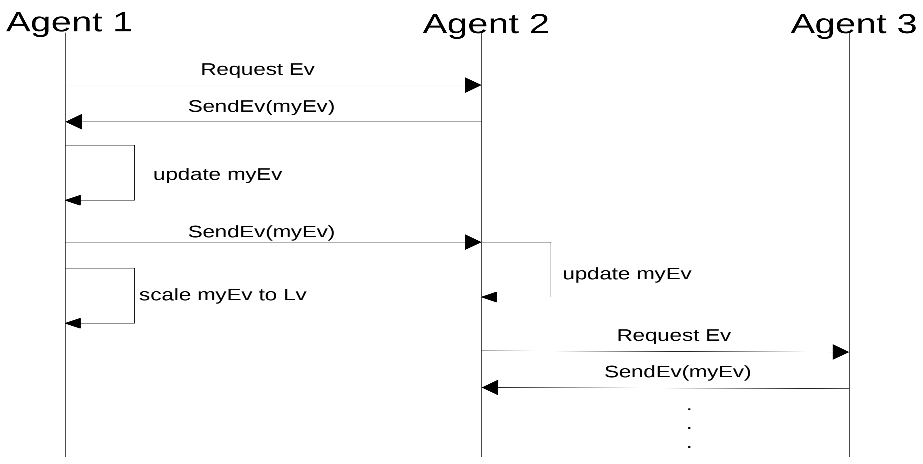

Fig. 3.7: ***Consensus work flow*** *between* 3 *agents.*

Agent 2 as a partner and commences gossiping. Once complete, Agent 2 begins their gossiping interaction, randomly selecting Agent 3 to interact with.

3.4.4 DETect and Design Objectives

Through a combination of sliding observation windows, statistical analysis using MLR and CUSUM and distributed consensus, DETect has been designed to address all the objectives for decentralised emergence detection outlined in Section 3.1. This is summarised in Table 3.1, where specific features of DETect are mapped to each design objective. DETect removes the need for a centralised observer of the global system, with DETect instead using the constituent agents as detectors (objective 1). The agents use information that is already available to them, characterised by a set of internal and external variables, to facilitate emergence detection. DETect allows agents to act as detectors at runtime by reducing the amount of design-time information required by the agent about the emergent properties of the system (objective 2). DETect also uses an automatic model selection mechanism, LASSO, to select the relationship between internal variables and external variables that should be monitored (objective 5). To increase the scope of information an agent receives from the system, DETect uses a gossip-based dis-

Table 3.1: Design Objectives and DETect Features

Obj.	Description	DETect feature
1.	Decentralise detection	Constituent agents as detectors, Distributed Consensus
2.	Detect emergence at runtime	Constituent agents as detectors, LASSO model selection, CUSUM change detection
3.	Use only locally available information	MLR modelling of internal and external variables
4.	Support collaboration between agents	Distributed gossip-based consensus
5.	Autonomously select data to monitor	LASSO model selection
6.	Detect formation and evaporation of emergence	Sliding observation windows, CUSUM change detection

tributed consensus mechanism. This distributes the detection of emergence across all the agents in the system, instead of having a single agent hold responsibility (objective 4). Finally, DETect constantly updates its model of the agent's relationship with its environment by using a series of sliding observation windows, allowing transient changes in the emergent system state to be detected (objective 6).

3.5 Limitations Due to Design Decisions

A number of the decisions made when designing DETect have introduced some associated limitations to the DETect algorithm. Although it is not believed that these limitations significantly undermine the feasibility of DETect's approach, these limitations are discussed in the remainder of this section. A detailed evaluation of the performance and feasibility of DETect is presented in Chapter 5.

Multiple Linear Regression: The modelling technique used by DETect is multiple-linear regression (MLR) where internal variables are modelled using the external variables as explanatory variables. Using MLR means non-linear relationships that may exist between an internal variable and one or more external variables may not be modelled entirely accurately. This is because these relationships will be approximated to a linear model resulting in some information, or subtlety in the relationship, being lost.

In the context of DETect, the purpose of the model is **not** to facilitate prediction of the internal variable, where such subtlety could greatly impact the accuracy of predictions made.

Instead, the purpose of the MLR is to provide a consistent measure of the relationship, not to achieve a fully accurate understanding of how the two variables relate to one another. In other words, *DETect requires a measure that behaves consistently over time*, enabling a baseline of expected p-values for the relationships to be established, which can then be used by the CUSUM algorithm. For example, if the MLR is a bad model of a given relationship and the p-values generated vary wildly over time, this is acceptable for DETect's purpose so long as the p-values continue to act in this manner. In this instance, the CUSUM is designed to detect when this behaviour changes, for example, if the p-values suddenly begin to remain static (the standard deviation will change) or trend in a certain direction (the mean will change).

It is possible that non-linearity could be introduced into the MLR by the inclusion of interactions between explanatory (external) variables. For example, external variables could be multiplied with one another or themselves to create a new set of variables. This technique is typically used when manually selecting a model and involves many iterations to find meaningful interactions. However, with automated model selection and with no limit on the number of potential external variables or combinations possible this would greatly increase the computational cost of model selection and renders the use of this technique in DETect impractical.

Another possibility when using MLR is that p-values may vary due to random chance when no real change has occurred in the system. In this instance, DETect's CUSUM algorithm may detect a change and a false positive feedback detection event may be generated. DETect's consensus algorithm is designed to address this scenario as a single agent experiencing a change will not be able to form consensus with other agents. The possibility of a sufficient number of agents simultaneously generating similar false positives does exist in theory, however the likelihood of this happening is extremely small.

In summary, MLR provides an efficient means of generating a relationship measure for multiple variables at once. It is possible that some relationships may not be modelled entirely accurately, as a result. However, since the objective of DETect is not a predictive one, and due to robustness provided by both the CUSUM and consensus algorithms, these limitation are not considered to fundamentally undermine the use of MLR in this context.

Unchanging variable set One of the assumptions made during the design is that the set of internal and external variables available to the agent do not change during the system evolution (*cf.,* Section 3.2.3). This means that no new variables are added and none of the existing variables become unavailable. As a result, DETect's initialisation phase is only executed once for each agent during the system evolution. The set of available variables may change, for example, following a comprehensive adaptation of the agent or if the environment changed fundamentally. The impact of such scenarios are not considered as part of this work.

Model Selection The model selection algorithm is repeated every time the sliding observational windows are full until at least one internal-external variable pair is selected to comprise the model. It is assumed that eventually one relationship will be selected by DETect's modelling unit. However, it is theoretically possible that it may take a prohibitively long time for this to happen on one or more agents, or indeed it may never happen. In such a scenario, the remainder of DETect's work flow, feedback detection and consensus formation, would not be utilised on that agent. Instances where an agent fails to achieve this minimum requirement are not considered and the scenario did not arise at any time during evaluation.

3.6 Summary

This chapter outlined the design objectives necessary to meet the challenges presented by emergence detection in Complex Adaptive Systems. Next, the terminology and system model that are the subject of this thesis were outlined to identify the scope of and assumptions made by this work. The subsequent sections described the design decisions motivated by the identified design objectives and described the solution proposed by this thesis.

The proposed solution for emergence detection, DETect, addresses these design objectives using a combination of sliding observation windows and statistical analysis. The configuration of these techniques is designed to allow the constituent agents of the system to detect feedback from emergence through changes in their statistical relationship with their environment. Once feedback is detected, a distributed consensus algorithm is used to enable agents to share information and therefore base their conclusion of emergence on more than their own limited experience.

Chapter 4

Implementation and Simulation Environment

This chapter describes the implementation of DETect and the simulation environments that are used for evaluation. It begins by introducing NetLogo, a multi-agent programmable modelling environment that is used to implement DETect agents and perform experimentation. Extensions to NetLogo were implemented in Java and R which contain the core functionality provided by DETect. Next, the implementation of a generic DETect agent is outlined before the three multi-agent models implemented as case studies to evaluate DETect's performance in Chapter 5 are described. The interface between agents and their environment and the behavioural rules that are the foundation of each model are outlined.

4.1 Simulation Framework

Figure 4.1 illustrates the components of the simulation environment with DETect components highlighted in green. DETect's implementation is distributed across NetLogo, Java and R.

NetLogo

NetLogo v5.1.0 was used for the development of the simulation environments [Wilensky, 1999]. NetLogo is an open source programmable modelling environment that is written in Java and runs on the Java Virtual Machine (JVM). It allows users to write their own extensions in Java

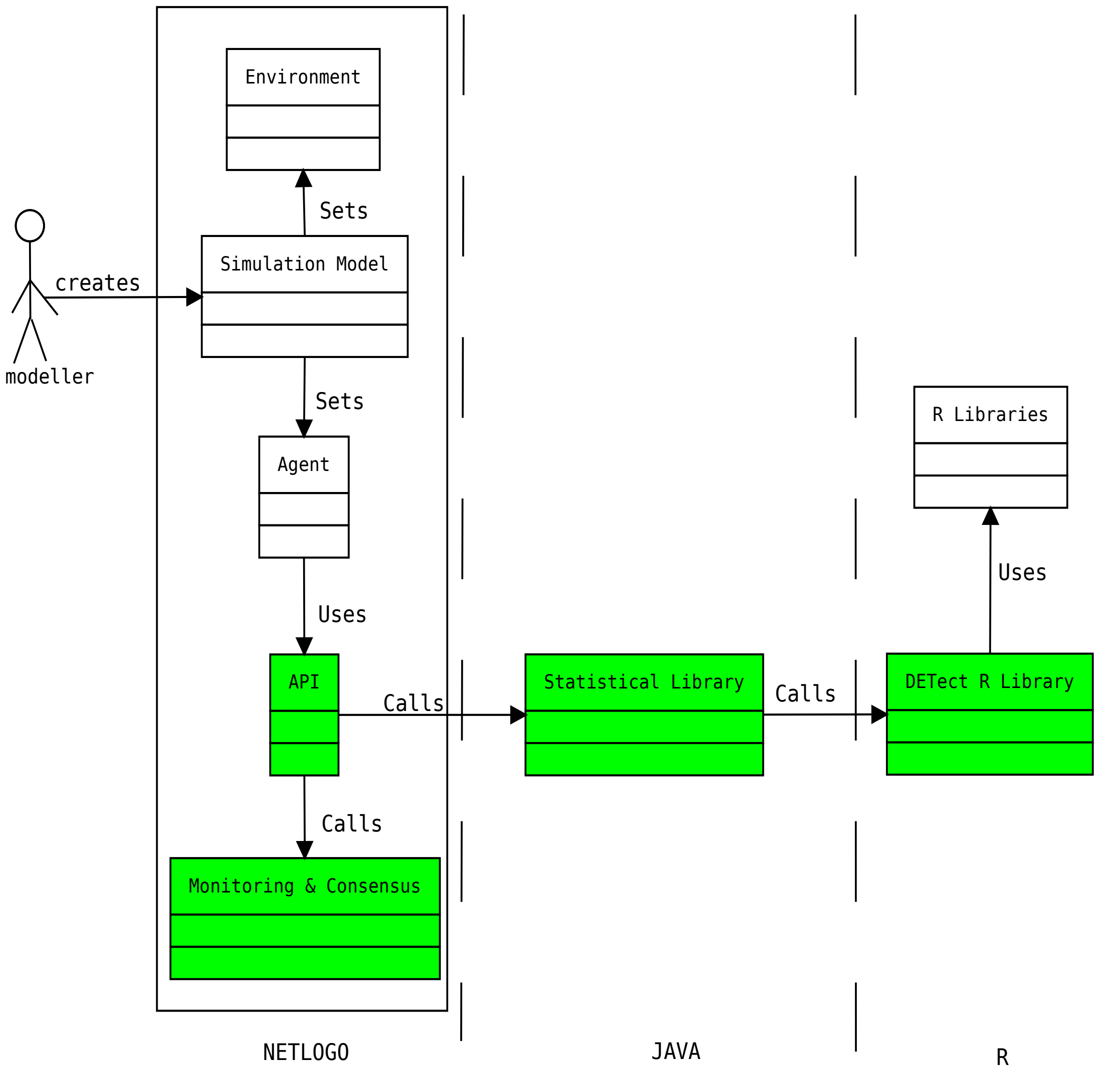

Fig. 4.1: ***Simulation Framework:*** *DETect components, highlighted in green, are implemented across NetLogo, Java and R.*

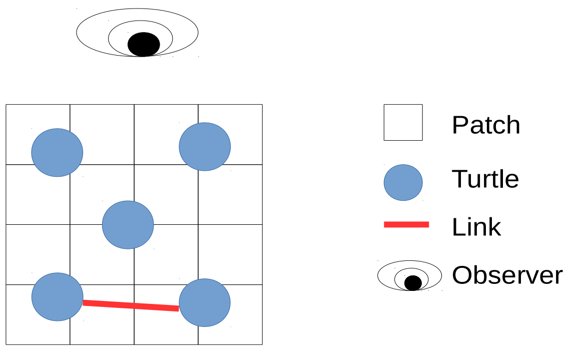

Fig. 4.2: ***NetLogo Agent Types:*** *Patches create the two dimensional grid of the simulation providing turtles with a world to move around on. Links can be used to connect two turtles together. The Observer monitors the simulation and can give instructions to the other agents.*

that can be loaded and run during simulations via an API. It has been used extensively in the study of emergence in multi-agent systems, modelling a variety of domains such as disease spreading [Tang and Mao, 2014], flocking [Chan, 2011] and pedestrian counter flow [Procházka and Olševičová, 2015].

Simulations in NetLogo are described by a *simulation model*, written in NetLogo's own modelling language. This is used by the simulation engine to specify the environment and agents that should be created. NetLogo contains 4 types of agents illustrated in Figure 4.2. They are introduced here and used to describe the 3 case studies later in the chapter. The agent types are:

- **Patches** are stationary agents. They form a two dimensional grid that makes up the coordinate system of the world.
- **Turtles** are agents that move around in the world. It is possible to create different breeds of turtle, each with unique parameters and behaviour rules.
- **Links** are connections that are made between two turtles.
- **The Observer** agent is a super agent that can be used by the modeller to give instructions

to agents or to monitor the properties of agents in the system. The Observer advances time in the simulation

The simulation model also contains the behavioural rules of the agents and the environment, which contain details of how agents move and interact with one another. Sections 4.3 to 4.5 below describe in detail the models used in each of the 3 case studies. Agents of type *turtle* use DETect, as their ability to move in the environment allows them to generate the spatial emergent behaviour that is the focus of this thesis. Therefore, from this point on, the term *agent* is used to refer to turtles, while patches, the observers and links are referred to explicitly by name.

DETect API

Agents in the simulation can access DETect functionality through DETect's API that exposes the statistical, monitoring and consensus algorithms to the agent.

Monitoring and Consensus

The monitoring and consensus aspects of DETect are implemented in NetLogo's modelling language. The Monitoring functionality refers to the initialisation and variable observation (*cf.,* Section 3.4.1.1), where the averaging and sliding observation windows for all internal and external variables are maintained. The Consensus algorithm (*cf.,* Section 3.4.3) is responsible for distributed consensus formation about the presence of an emergent event in the system.

DETect Statistical Library

The more complicated statistical functionality provided by DETect required a separate JAVA library to be implemented. This functionality includes both the model selection (*cf.,* Section 3.4.1.2) and model analysis (*cf.,* Section 3.4.1.3) algorithms described in Chapter 3. The library is loaded by the JVM when NetLogo is started and its functionality can be called by agents from within the NetLogo simulation environment at runtime. When each agent is initiated, a DETect object is created for them that stores the internal and external variables for that agent and tracks the history of the agent's relationship with its environment as the system evolves. In addition, this object passes data into R and executes functionality written in R to analyse the data. The methods exposed to the agent by this library are as follows:

- **void initiateDETect(String agentName)**, which is used to create a DETect object used to manage the agent's internal and external variables and monitor its relationship with its environment.

- **void addVariable(String varType, String varName)**, which is used to add a new variable for DETect to model. The variable is given a type, either "internal" or "external", and assigned a name.
- **void updateVariable(String varType, String varName, double[] slidingObservationWindow)**, which is used to update a variable's values in DETect once its sliding observation window is full.
- **int runLasso()**, which is used to initiate the LASSO-based model selection. This results in a model composed of internal-external variable pairs, where a multi-linear regression (MLR) is applied to each internal variable and all its associated external variables. An integer value is returned indicating how many internal variables are part of the model. If at least 1 internal variable is selected, model selection is deemed a success.
- **void runRegression()**, which is used to trigger an analysis of the selected model for all internal variables and their associated external variables using MLR. This function dynamically creates the correct statements to load the variable data and perform the MLR in R.
- **double[] runCusum(String varName)**, which is used to initiate a Cumulative Sum (CUSUM) analysis of the internal-external relationships included in the model for the specific internal variable, **varName**. An array is returned indicating the CUSUM for each external variable that is included in the internal variable's model. This is used to determine if any of these relationships has changed.

DETect R Library

The final component in the implementation is a dedicated R library containing aspects of the model selection and model analysis algorithms. This functionality is called from the DETect statistical library with the R library *glmnet* used to perform the LASSO analysis. The functions included in DETect's R library are:

- **chooseVariables(internal,external)**, which implements Algorithm 2 (*cf.,* Section 3.4.1.2). It takes the set of internal and external variables for an agent and returns the names of each internal-external variable relationship selected by the algorithm. This is called as part of the **runLasso** routine described above.

- **injectNoise(slidingObservationWindow)**, which is used to inject zero mean noise to the data in preparation for MLR and LASSO.

4.2 Generic DETect Agent Implementation

This section presents the implementation of a generic DETect agent. The steps involved during agent initialisation and a generic time step in a given simulation are described in detail, showing how control and information flows through the components in the simulation framework. The steps described for both of these processes are common across all models where DETect is implemented.

4.2.1 Agent Initialisation

Figure 4.3 illustrates the sequence of steps involved in the initialisation of an agent using DETect. This sequence corresponds to the initialisation algorithm described in Section 3.4.1.1.

- **Step 1-2** The sequence begins with the *simulation model* where the environment and agent are specified and initialised.
- **Step 3-4** The agent requests that a DETect object is initialised using the DETect API. This involves a call to the DETect Java statistical library where a new DETect object is created and assigned to the agent.
- **Step 5-7** Next the agent informs DETect of its set of internal and external variables. A sliding observational window is created for each variable in DETect's monitoring and Consensus component and the data structures for each variable are also created using DETect's statistical library.
- **Step 8-9** Once complete, control is passed back to the environment. The clock is updated to the next time step and the agents are instructed to move. The move an agent makes is determined by its behavioural rules, which are unique to each simulation model. Section 4.3 to Section 4.5 describe the models implemented in the simulation environment and outline the behavioural rules used by agents in each model.

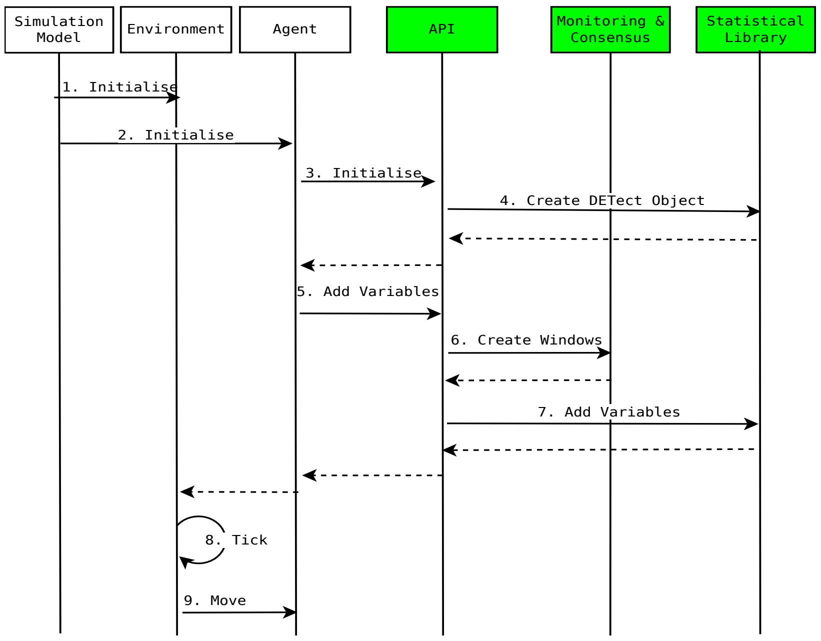

Fig. 4.3: ***Generic DETect agent - Initialisation Sequence Diagram***

4.2.2 Generic Time Step

Once the environment and the model have been initialised the system will evolve with each agent moving and interacting at each time step. Figure 4.4 describes the sequence of steps executed during each time step, demonstrating how information flows into DETect. This sequence is more complicated than the initialisation sequence as there are two decision points depending on whether the sliding observation windows are full and whether the model has been selected. The steps involved in this sequence are:

- **Step 1-3** The sequence begins with the environment advancing the system clock and instructing each agent to move. On receiving this prompt, the agent makes a decision on its action based on its behavioural rules.

- **Step 4-5** Next, an observation is made of the agent's internal and external variables and the observations are passed into DETect, where the *Monitoring & Consensus* component add

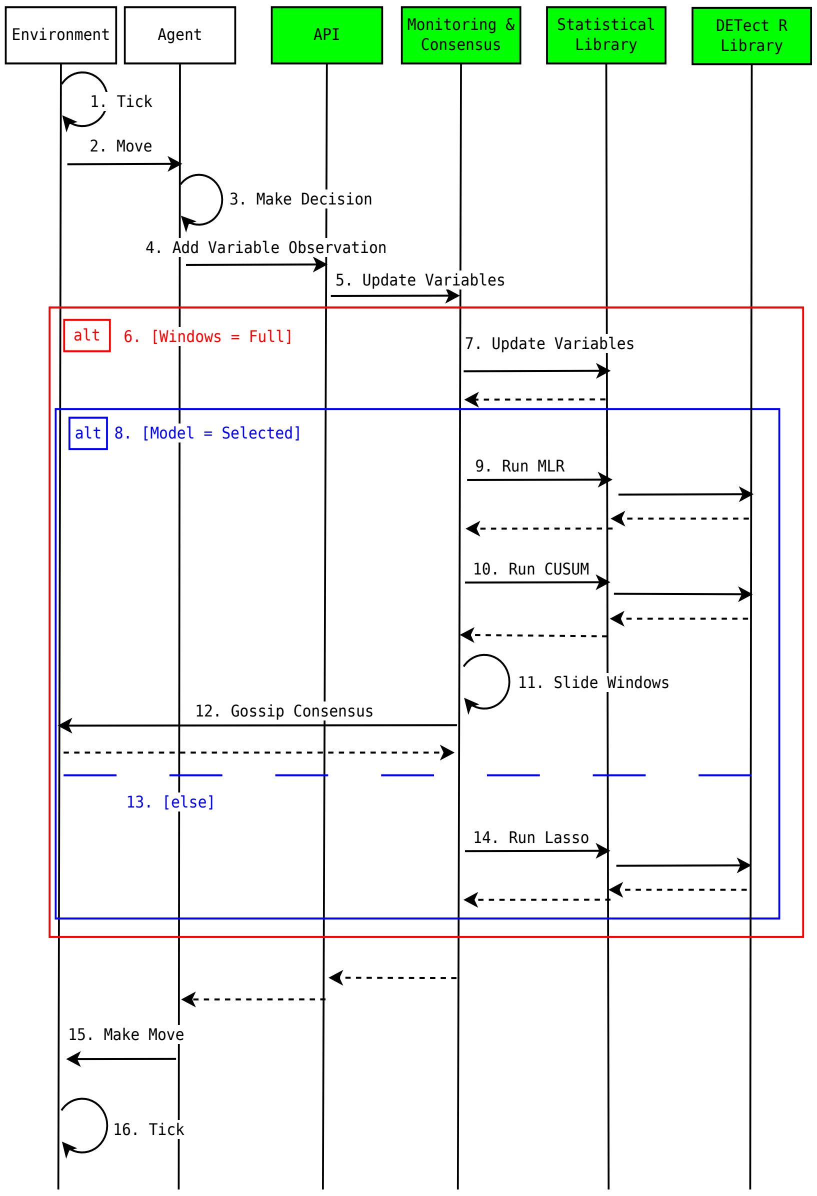

Fig. 4.4: ***Generic DETect agent - Activity Sequence Diagram***

them to their respective sliding observational windows (*cf.,* Algorithm 1, Section 3.4.1.1). The internal and external variables for each model are outlined in Section 4.3 to Section 4.5.

- **Step 6-7** If the windows are full, they are forwarded to DETect's statistical library. If the windows are not full, the sequence jumps to Step 15-16. Assuming the windows are full, another IF-ELSE decision is made. If the model of the agent's relationship with its environment has already been selected Steps 8-12 are executed. Else, if the model has not been selected, Steps 13-14 are executed.
- **Step 8-12** The internal-external variable model has already been selected so these steps proceed to analyse it using an MLR. The p-values generated by the MLR are returned and forwarded to the CUSUM algorithm to determine if a significant change has occurred in the relationship. Once this is complete, the results are returned to the *Monitoring & Consensus* component and the sliding windows are progressed by deleting the earliest values. Next, the gossiping consensus protocol is executed with the environment searched to find a gossiping partner. Once complete, the process continues to Step 15-16.
- **Step 13-14** (From Step 6-7) The LASSO-based model selection algorithm is executed to select the internal-external variable relationships that will be used to model the agent's relationship with its environment.
- **Step 15-16** Control is returned to the agent, who makes the move or action decided upon. Once all agents have moved, the environment advances the system clock.

4.3 Model 1: Flocking

Boid flocking is a well-established model used in the study of emergence [Reynolds, 1987]. The emergence in this model occurs when patterns form in the aggregate behaviour, such that the flock appears to move in unison. When implementing the model, each boid is represented as an autonomous agent, with the formation of flocking patterns achieved through the local interactions between these agents. These interactions follow simple rules to update each agent's speed and heading:

1. **Separation**: Steer away from other agents that are too close.

Table 4.1: Parameters used to control flocking behaviour

Rule	**Description**	**Parameter Name**	**Generate Flocking**	**No Flocking**
Align	Adjust heading so that you steer towards the average heading of your flockmates	A	5	0
Cohere	Adjust heading so that you steer towards the center point of your flockmates (average (x,y) coordinates)	C	3	3
Separate	Adjust heading so that you steer away from your nearest flockmate	S	1.5	1.5

2. **Alignment**: Adjust heading and speed so that the agent is moving in the same direction and at the same speed as agents nearby.

3. **Cohesion**: Steer towards the centre of other agents nearby.

4.3.1 Behavioural Rules

The simulation uses the standard flocking model found in the NetLogo modelling commons [Wilensky, 1998]. It contains 150 DETect enabled agents in a squared environment, sized 50x50 patches, where the width of a patch corresponds to 1 unit of distance in the world. The environment wraps both horizontally and vertically allowing agents that fly through any border of the environment to appear again seamlessly from the other side. The (X,Y) coordinate system in the environment is based on the patches that make up the world, with the center patch representing point (0,0). The heading of an agent, its direction of travel, is described in degrees ranging from 0 to 360, similar to a compass. A heading of 90 degrees indicates the agent is moving left to right (directly due east) and 180 degrees means the agent is moving from top to bottom in the world (directly due south).

At each time step, all agents update their heading and speed using the three rules outlined above before moving forward in their new direction. The rate at which agents adjust their heading in response to each rules is controlled using 3 global parameters specifying how many degrees to adjust their heading by. The values used for each of these rules during simulations are outlined in Table 4.1. Flocking behaviour is controlled during the simulation by adjusting the number of degrees an agent aligns its heading with its flockmates.

Figure 4.5 shows the field of vision of each agent in the model. Each agent's "vision" is a

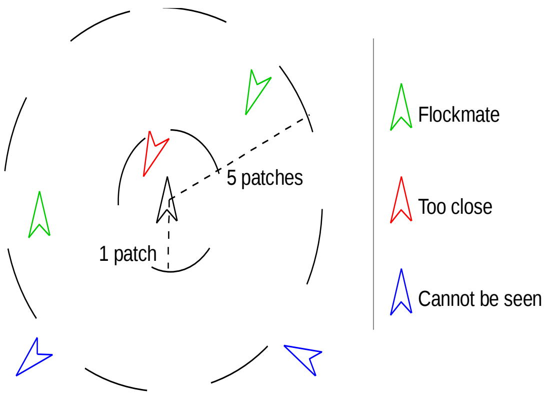

Fig. 4.5: ***Flocking agent's field of vision***

distance of 5 patches from their center point. This means that agents see only other agents within this radius (the outer circle). Agents apply their behavioural rules based on the agent set found inside this threshold, so the blue agents are ignored. Agents inside this radius constitute the agent's "flockmates". An agent will deem another agent to be too close when they are within 1 patch of their own position (the red agent inside the inner circle), triggering the agent to steer away from this agent. Despite this, it is possible for agents to appear to inadvertently "collide" during the simulation. Figure 4.6 presents a flowchart of this flocking process that is executed by agents at each time step.

Speed is also updated during this process. All agents are initialised with a common speed of 1, indicating the number of patches they traverse at each time step. During the simulation, each agent tries to match the average speed of their flockmates if there are any other agents nearby. The process concludes with each agent making a small autonomous acceleration or deceleration, selected randomly from a normal distribution with mean 0 and standard deviation of 0.1.

4.3.2 DETect Integration

Each agent monitors 4 internal and 6 external variables, described in Table 4.2, giving a total of 24 internal-external variable relationships for DETect to choose from. Among these are two internal (Age and Vision) and two external variables (temperature and Distance to Center) that are either static or randomly varying and are not used by the agent to perform an action. As a system modeller, these variables are deemed to be inconsequential to the model, however DETect is unaware of the semantic content of all variables and what constitutes variables associated with

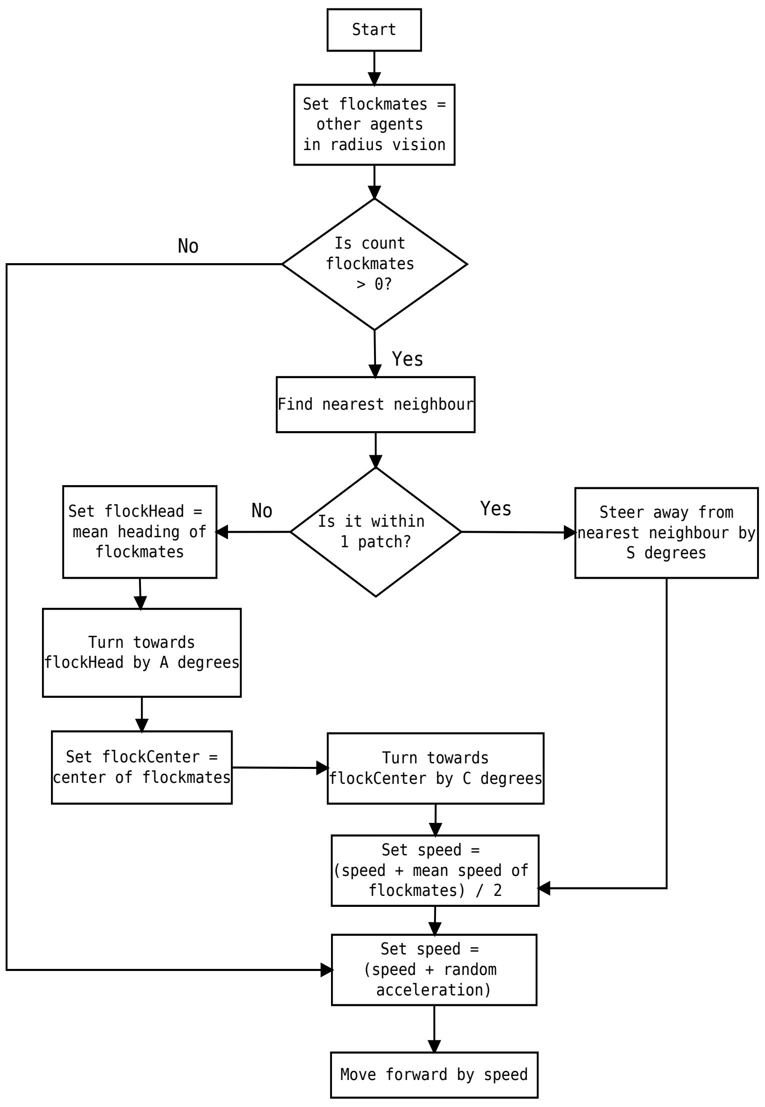
Start
Set flockmates = other agents in radius vision
Is count flockmates > 0?
No
Yes
Find nearest neighbour
Is it within 1 patch?
No
Yes
Set flockHead = mean heading of flockmates
Steer away from nearest neighbour by S degrees
Turn towards flockHead by A degrees
Set flockCenter = center of flockmates
Turn towards flockCenter by C degrees
Set speed = (speed + mean speed of flockmates) / 2
Set speed = (speed + random acceleration)
Move forward by speed

Fig. 4.6: ***Flocking behaviour flowchart***

Table 4.2: Flocking Internal and External Variables

Name	Description	Potentially Useful
Internal Variables		
Speed	The current speed of the agent	Yes
Heading	The current heading in degrees of the agent	Yes
Age	A static variable randomly selected at agent initialisation	No
Vision	How far the agent can see. Variable is static during simulation	No
External Variables		
Neighbours Heading	The mean heading agents in the agent's field of vision	Yes
Neighbours Speed	The mean speed agents in the agent's field of vision	Yes
Spacing	The distance to the closest other agent	Yes
Number of Neighbours	The number of agents in the agent's field of vision	Yes
Temperature	A randomly varying environment variable	No
Distance to Centre	How far the agent is from the center of the map	No

flocking and those that are not associated with flocking. DETect is only concerned with finding sets of variables where the relationship between them contains some information. As a result, variables that are deemed to be not potentially useful, are included in the implementation to enable the model selection process in DETect to be evaluated. This evaluation is described in more detail in the next chapter (*cf.,* Section 5.1). Finally, the flockmates of an agent at time t are used as gossiping candidates for DETect during the simulation.

4.3.3 Monitoring Emergence Objectively

The emergence to be detected in this model is the coherent flocking patterns that form when agents appear to move in unison. To measure when this flocking occurs and the extent to which it occurs, a centralised method similar to that described by Niazi and Hussain is used to provide an objective measure of emergence in the system [Niazi and Hussain, 2011]. This objective measure can then be used to identify times when emergence is forming and evaporating the system which corresponds to times when DETect should generate detection events.

The objective measure is based on the intuition that when flocking forms, agents get closer together, as illustrated in Figure 4.7. The image on the left contains no flocking behaviour and agents are close to uniformly dispersed across the simulation world. The image on the right does contain flocking and agents are bunched together with large areas of the world empty. Therefore, each patch that makes up the simulation environment is equipped with a proximity sensor that

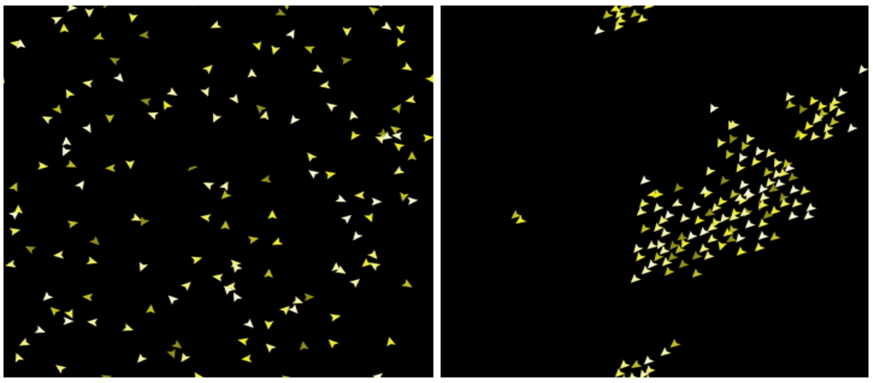

Fig. 4.7: ***Flocking model screen shot:*** *A multi-agent system simulation of boids demonstrating no emergent behaviour (left) and emergent flocking behaviour (right).*

indicates if any agent is within a radius of 2 patches from its center. The percentage of patches with no agents within this radius is recorded every 50 time steps throughout the simulation. Once the simulation is over, the time series of this measure can be used to identify periods with flocking and periods without flocking. As a result, transitions between these periods corresponding to emergence formation and evaporation can also be identifed. The process used to identify these transition periods is described in Appendix A.

4.4 Model 2: Pedestrian Counter Flow

The second simulation is of pedestrian agents, another well-studied model in the field of emergence and self-organisation [Helbing and Molnar, 1998, Helbing et al., 2002, Moussaïd et al., 2011]. There are many examples of possible emergent behaviour in pedestrian models, such as congestion points, however this study focusses on emergence characterised by the formation of lanes in counter-flowing individuals. An existing pedestrian model previously described in the study of information entropy during the formation of these emergent behaviours [Procházka and Olševičová, 2015] is used. The core behavioural aspects of the agents are outlined in the next section.

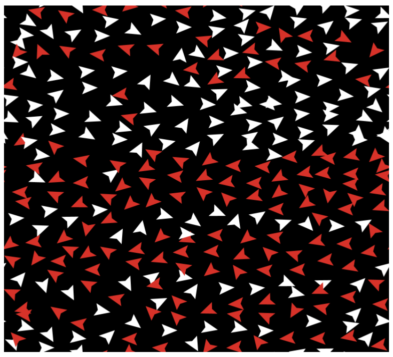

Fig. 4.8: ***Pedestrian model screen shot:*** *Illustrating the counter-flow behaviour in the pedestrian model.*

4.4.1 Behavioural Rules

The simulation, illustrated in Figure 4.8, uses 380 agents in a squared environment (sized 40x40 patches) that wraps both horizontally and vertically. The agents are split evenly into two groups, red and white. Each agent has two types of behaviour, a random walk and a guided walk. The guided walk behaviour involves counter-flow, with the white agents attempting to travel from left to right (heading = 90) and the red agents attempting to travel in the opposite direction (heading =270). Each agent's behaviour can be summarised by three rules: (1) avoid collisions with other agents (2) move to less crowded spaces and (3) maintain the desired heading (either 90 or 270 degrees).

Every time step, an agent will attempt to update its heading to satisfy the three specified rules. It does this by finding an un-crowded patch within a 2 patch radius of itself. The patches within this radius constitute the agent's candidate patches. To achieve this, all agents tell their candidate patches how far they are from the patch's center. This distance is inverted by subtracting it from the maximum possible distance of 2 patches. Once each agent has communicated with their candidate patches, each patch sums the inverted distances creating a measure of how crowded they are, called a crowded coefficient. When agents wish to select from amongst their candidate patches, they select the patch with the lowest crowded coefficient that requires the smallest deviation from its existing heading. Once complete, the agent turns towards the chosen patch and moves forward. In the model, each agent's speed is maintained at 0.3 patches per time step.

Table 4.3: Pedestrian Internal and External Variables

Name	**Description**	**Potentially Useful**
Internal Variables		
Heading	The current heading in degrees of the agent	Yes
Speed	The current speed of the agent. This is static throughout.	No
Age	A static variable randomly selected at agent initialisation	No
Height	A static variable representing the agents height	No
Weight	A static variable representing the agents height	No
External Variables		
Neighbours Heading	The mean heading agents in the agent's field of vision	Yes
Neighbours Speed	The mean speed agents in the agent's field of vision	Yes
Spacing	The distance to the closest other agent	Yes
Number of Neighbours	The number of agents in the agent's field of vision	Yes
Temperature	A randomly varying environment variable	No

4.4.2 DETect Integration

Each pedestrian agent monitors 5 internal and 5 external variables, outlined in Table 4.3. This gives a total of 25 internal-external variable relationships for DETect to choose from. However, four of the internal variables, speed, age, height and weight are static and are not used by the agent when selecting an action. Similarly, the external variable temperature, is a common variable across all agents and varies randomly throughout the simulation. As a result, temperature is labelled as not useful. Finally, agents in the model gossip with other agents within a 5 patch radius when forming consensus on the presence of emergence.

4.4.3 Monitoring Emergence Objectively

The emergent behaviour of interest in this model is the formation of lanes between agents in the environment. Lanes are formed by individual agents who follow each other in a queue that can be clearly distinguished. Procházka and Olševičová [Procházka and Olševičová, 2015] formally define a lane element as a triplet of agents travelling in the same direction, who meet the following constraints:

- **Closeness** - the distance from the middle agent to outer agents of the triplet is within the defined interval <lane_radiusmin, lane_radiusmax>
- **Straightness** - the angle between two lines from the middle agent to the outer agents of

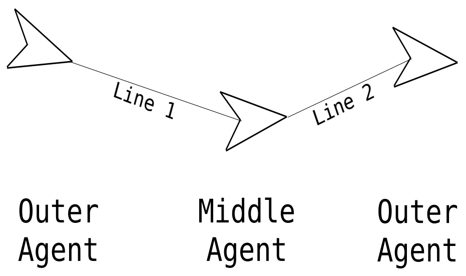

Fig. 4.9: ***Pedestrian lane triplet***

the triplet is within the defined interval <lane_anglemin, π >

- **Clearness** - there are no other agents between the middle and both outer agents of the triplet

Figure 4.9 illustrates how these constraints operate in practice. Line 1 and Line 2 illustrate the distance from the middle agent to both outer agents. The length of both of these lines must be greater than a minimum, set to 1 patch, and less than a maximum, set to 5 patches, to satisfy the Closeness constraint. Next, the angle formed by Line 1 and Line 2 when they meet at the middle agent must be less than π (180 degrees, a straight line) and greater than a minimum angle, set to 100 degrees. Finally, no other agents can be in the space between the middle and the two outer agents.

The model requires that at least 4 such lane elements exist contiguously for a full lane to be formed. The number of full lanes present in the system is counted throughout each simulation creating a time-series once the simulation is over. A high number of lanes indicates emergence is present in the system and identifying these periods allows the formation and evaporation periods of emergence to be found. Appendix A describes the process used to identify these transition periods in the time series of the objective measure.

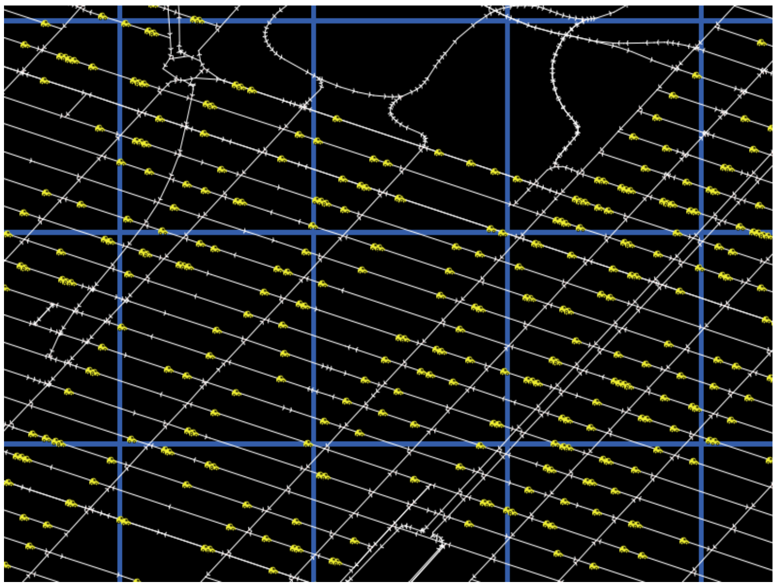

Fig. 4.10: ***Traffic model screen shot:*** *Showing a segment of the street network (white), taxi agents (yellow) and post-code borders(blue).*

4.5 Model 3: Traffic

The final simulation is of road traffic, where OpenStreetMaps is used to model the road network of Manhattan, New York City [BBBike.org, 2014]. Agents are implemented as taxis designed to traverse a model of the Manhattan street network with GraphHopper [Graphhopper, 2014]. GraphHopper is an open source routing library, based on OpenStreetMap data, used to ensure realistic routing of agents. A screen shot of the model is provided in Figure 4.10 with the agents visible as yellow on the map. The blue boxes represent zip-code areas, each containing a taxi-depot where the agents begin the simulation. Agents are initially evenly distributed across the map and they traverse the map with the goal of travelling to all depots during the simulation.

The simulation uses 2500 taxi agents, with each agent attempting to complete their journey while avoiding collisions with other agents and stopping for traffic lights at intersections. Agents travel from depot to depot around the map with each depot initially being selected randomly by each agent. Periodically, all agents are directed to choose the same depot as their next destination, forcing all agents to converge to the same area of the map. This convergence leads to the formation of a traffic jam which is the spatial emergent behaviour of interest in this model.

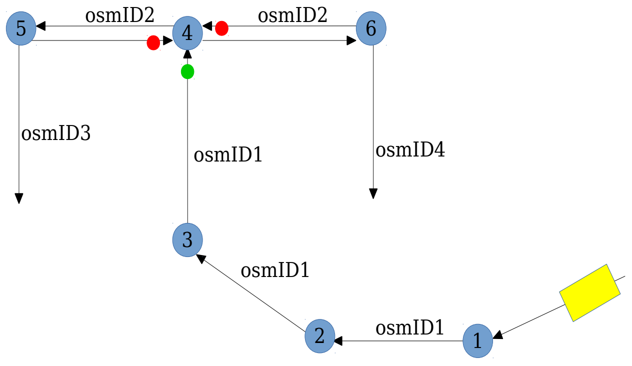

Fig. 4.11: ***Example street network*** *in the traffic model with junctions in blue and streets connecting them. The taxi agent is displayed in yellow.*

4.5.1 Street Network and Navigation

The street network is constructed as a directed graph as illustrated in Figure 4.11. Each street is identified with a unique OSM id and the arrow indicates the permissible direction of travel. Both one-way and two-way streets are implemented. The network is constructed by dividing the street into a series of straight line segments and placing a junction, the blue nodes, where these segments meet. This allows curves in the same street to be constructed using a series of connected junctions, as demonstrated with *osmID1* using junctions 1, 2 and 3.

Junction 4 represents the convergence of three different streets. When the taxi agent (yellow) arrives at the junction it must decide which street to follow. Each agent maintains a list of the OSM ids it must follow in turn to reach its destination. Each junction maintains a list of the other junctions connected to it and a list of the OSM ids that are available if the agent travels in the direction of that junction. So, for example, Junction 4 can inform the agent that if it travels towards Junction 5 it can reach *osmID3* and if it travels towards Junction 6, it can reach *osmID4*. The agent will then adjust its heading to face towards the junction that offers it the next street on its journey.

Finally, as the streets coming into and leaving Junction 4 present potential conflicts, a set of traffic lights is assigned to each street indicating if agents approaching on that street can use the

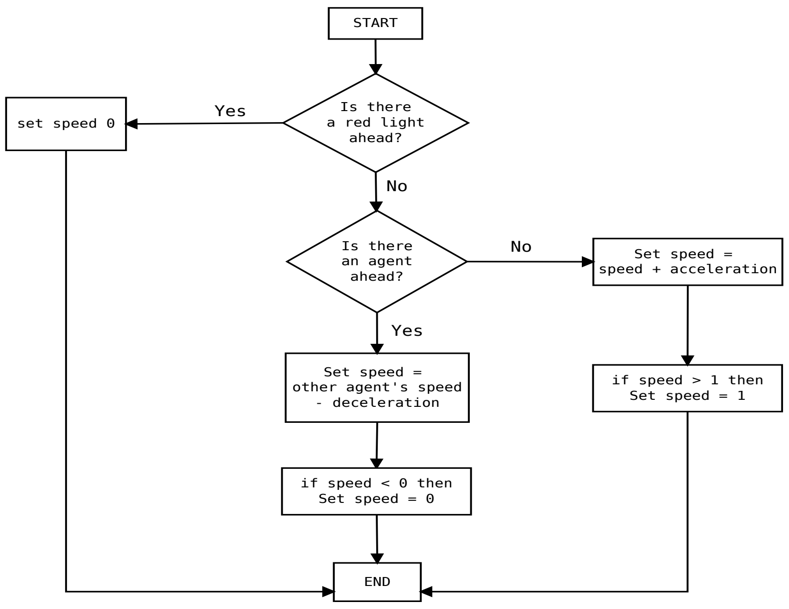
START
Is there a red light ahead?
Yes
set speed 0
No
Is there an agent ahead?
No
Set speed = speed + acceleration
Yes
Set speed = other agent's speed - deceleration
if speed > 1 then Set speed = 1
if speed < 0 then Set speed = 0
END

Fig. 4.12: ***Decision tree for speed updates:*** *The agent looks for traffic lights and other agents ahead of it.*

junction. A round robin system is used to periodically update the traffic lights associated with the same junction ensuring that each approaching street is assigned a "green light" in turn. All junctions that have two or more streets flow into them are assigned a set of traffic lights when the model is initialised.

4.5.2 Behavioural Rules

At each time step an agent tries to move towards its destination. This involves two steps, updating their speed and moving.

Update Speed

Agents travel at a varying speed with a range of 0 to 1 patch per time step and have a fixed vision of 1 patch in front of them. Unlike their heading, speed is updated constantly based on what is around them in the environment. Figure 4.12 illustrates the steps each agent takes to update their speed at each time step. First, the agent checks to determine if there is a red light for the street they are currently on within 1 patch distance ahead. If so, the agent immediately stops and waits for the light to change. If there is no red light, the agent next checks for other agents ahead travelling in the same direction on the same street as them. On two-way streets agents ignore other agents travelling in the opposite direction. If there is another agent ahead that is travelling along the same street segment, the agent slows down by setting its speed to the other agents speed, minus a deceleration parameter of 0.1 patch. If there is no other agent ahead, the agent increases its speed by accelerating by 0.11 patches. The final step is to ensure that the agent's speed is within the acceptable range of 0 to 1.

Movement

Each agent's direction of travel is determined by the next junction they are trying to get to, with their heading maintained to ensure the agent is pointed directly at the junction. The decision process used for agent movement is illustrated in Figure 4.13. If the agent is not at a junction, their movement is simply determined by their current speed. However, if the agent is at a junction there is a series of checks the agent must make before it can move through the junction. These decisions occur after the agent has updated their speed and movement is only possible if their speed is greater than 0. When there are no longer any other agents ahead in the queue, the agent may try to move through the current junction. However, this is only permitted if there is sufficent free space, 0.05 patches, after the junction on the street they intend to travel onto. If there is insuffiicent space available, the agent will wait in its current location. The reason this space may not be available is because there is congestion on the next street and there is no more capacity.

4.5.3 DETect Integration

Table 4.4 presents the internal and external variables monitored by agents in the traffic model. Each agent has 3 internal and 5 external variables giving a total of 15 internal-external variable relationships. As with the other models described in this chapter, these sets of variables contain static and randomly varying variables that are included to facilitate the evaluation of DETect's model selection algorithm (*cf.,* Section 5.1).

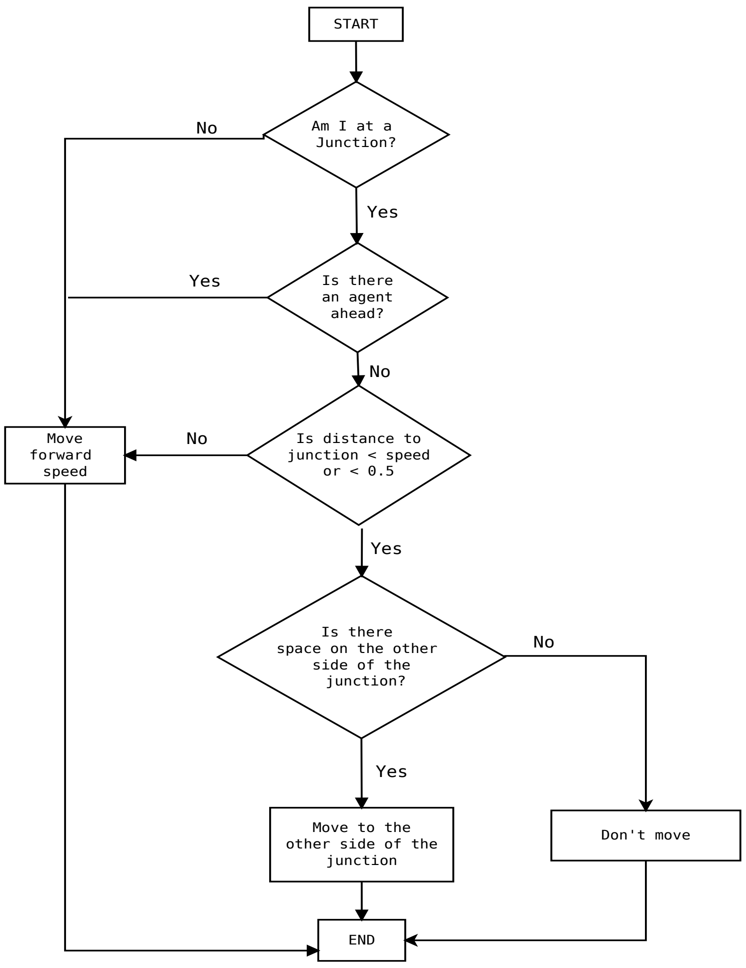
START
Am I at a Junction?
No
Yes
Is there an agent ahead?
Yes
No
Is distance to junction < speed or < 0.5
No
Move forward speed
Yes
Is there space on the other side of the junction?
No
Yes
Move to the other side of the junction
Don't move
END

Fig. 4.13: ***Decision tree for movement:*** *The agent's primary concerns are whether it is at a junction and its proximity to other agents.*

Table 4.4: Traffic Model Internal and External Variables

Name	**Description**	**Potentially Useful**
Internal Variables		
Heading	The current heading in degrees of the agent	Yes
Speed	The current speed of the agent.	Yes
Age	A static variable randomly selected at agent initialisation	No
External Variables		
Neighbours Heading	The mean heading agents in the agent's field of vision	Yes
Neighbours Speed	The mean speed agents in the agent's field of vision	Yes
Spacing	The distance to the closest other agent	Yes
Number of Neighbours	The number of agents in the agent's field of vision	Yes
Temperature	A randomly varying environment variable	No

4.5.4 Monitoring Emergence Objectively

As with the other two models described, a centralised objective measure of emergence is used to identify when emergence is present in the system. This objective measure provides a benchmark against which DETect can be evaluated. Each street segment that is greater than 2 patches in length is provided with a *manhole* sensor that is positioned halfway along its length. If a street is two-way, two manholes are used to monitored traffic travelling in both directions. The manhole is tasked with monitoring the number of agents that pass by and their average speed. The manhole can see agents in a radius of 2 patches of its center. The time taken for the agent to leave the manhole's field of vision from when it entered is used to calculate its average speed.

$$agentsPerHour = numberOfAgents * 50 \tag{4.1}$$

Every 50 time steps, called observation periods, all manholes in the system calculate a heuristic that approximates flowrate of traffic over the manhole. One thousand time steps is used as a proxy for one hour, enabling the number of agents per hour to be calculated based on how many agents the manhole observed since the last calculation. This is described in equation 4.1. The manhole uses the average speed and agents per hour measurements from the current observation period and the observation period immediately prior to calculate the flowrate. It does this by first calculating a *speed metric*, equation 4.2. Next, a volume metric is calculated as the average number of agents passing as a proportion of the maximum number of agents that could be seen during 50 time steps, equation 4.3. Finally, these two metrics are combined using a weighted average as specified in equation 4.4.

$$\begin{aligned} speedMetric = (averageSpeed_{currentPeriod} * agentsPerHour_{currentPeriod}) + \\ (averageSpeed_{lastPeriod} * agentsPerHour_{lastPeriod}) \end{aligned} \tag{4.2}$$

$$volumeMetric = mean(agentsPerHour_{currentPeriod} + agentsPerHour_{lastPeriod})/300 \tag{4.3}$$

$$flowRate = (speedMetric + (volumeMetric * 15)/2) \tag{4.4}$$

The flowrate metric is designed to be close to zero when there is a high volume of agents passing with a low average speed. This is intuitively what would be expected when traffic congestion is high. The final step is for each manhole to confer with other manholes as the emergence behaviour to be observed is not localised to a single street segment. Therefore, each manhole that has observed at least 1 agent during the last observation period calculates the average flowRate of all other manholes within 10 patches of itself. If its own flowRate and the average of its neighbours is below 0.45, the manhole signals that it detects congestion in its neighbourhood.

The number of manholes signalling congestion is monitored throughout each simulation run, enabling the formation and evaporation of congestion to be identified. This count is also used during simulation runs to determine if gridlock has occured, which is indicated by a high number of manholes reporting congestion that remains constant over time. When this occurs, a global policy is applied at traffic lights on congested streets where waiting agents are prompted to take an alternative route to their destination. This allows agents to move again and congestion levels to ease, allowing DETect to be evaluated during these periods of emergence evaporation.

The parameters used in the above heuristic were selected following initial prototyping and experimentation to achieve a metric corresponding to the congestion visible in the simulation world.

4.5.5 Summary

In this chapter, the implementation of DETect was presented as defined by the design outlined in Chapter 3. The main purpose of this implementation is to provide an initial prototype for experimentation and evaluation of DETect using a variety of mobility based multi-agent systems. The implementation of DETect is distrubted across NetLogo, Java and R with control and information flowing across these components during the initialisation and evolution of the system. This chapter also described the 3 multi-agent models that will be used in the next Chapter 5 to evaluate the performance of DETect. The unique behavioural rules that agents obey in each model were outlined along with the variables that are monitored by DETect in each model. Finally, what constitutes emergence in each model, and how it is objectively measured during each simulation run, was also described.

Chapter 5

Evaluation

This chapter presents an evaluation of DETect to determine how well it addresses the overall the objective of emergence detection in complex adaptive systems. The evaluation tests the hypothesis (*cf.,* Section 1.4) that when emergence is forming or evaporating, the constituent agents will simultaneously experience a change in the statistical relationship between themselves and their environment. By sharing this experience, agents can collaboratively act as detectors of the emergent event. The fundamental requirement for DETect is to provide a decentralised technique that allows constituent agents to detect the formation and evaporation of emergence in the system. In achieving this, DETect should also autonomously select what properties of the agent and its environment should be monitored.

Chapter 3 described how the design of DETect addresses the identified design objectives. However, the success of DETect as a technique for emergence detection depends on its performance in each of its three core functions: *model selection, feedback detection and emergence detection through consensus formation between agents*. For both feedback detection and consensus formation, DETect contains a number of parameters that are unique to the DETect algorithm and can affect overall performance of emergence detection. Therefore, the evaluation will examine the combination of parameters that achieve the best results, as well as determining the overall performance of the DETect approach. The evaluation objectives can thus be stated as follows:

1. To determine if DETect's model selection algorithm can autonomously select features of the agent, and the environment, that both enables feedback detection to occur and does

not require all available variables to be included in the model.

2. To determine what parameter combination achieves the best performance for both feedback detection and consensus formation.

3. To evaluate the performance of DETect at the best parameter combination to determine if it achieves the goal of decentralised emergence detection.

5.1 Experimental Setup

To evaluate the performance of DETect, a case study of three multi-agent model simulations implemented in NetLogo is used [Wilensky, 1999]. These models are flocking, pedestrian counter-flow and traffic. The implementation of each these models, as well as the emergence that they exhibit, are described in Chapter 4. In each simulation model, all agents are autonomous and each has its own unique implementation of the DETect algorithm. The case study is divided into three stages, each focussing on one of the three core functions of DETect: model selection, feedback detection and emergence detection through consensus formation between agents.

5.1.1 General Setup

Table 5.1 describes the simulation setup in each of the three models. The number of agents varies across each model, allowing DETect to be evaluated at three different scales. Each simulation is periodically switched between emergence and non-emergence phases allowing emergent behaviour to both form and disperse during the simulation. The length of emergence phases is common across each model at $10,000$ time steps. The length of simulations in the Traffic model, as well as the time steps when emergence phases begin, is different compared to the Pedestrian and Flocking models. This is due to the larger scale of both the world size and number of agents involved in this model with a period of time required at the start of each simulation to allow agents to spread out.

Switching between emergence phases allows change periods to be identified at the threshold of each phase where DETect should both detect feedback and subsequently build consensus on the presence of emergence in the system. The method used to control these phases is described in Chapter 4 for each of the three models.

Table 5.1: Simulation details for each model

Model	Number of Agents	Simulation Length (Time Steps)	Non-Emergence Periods (Time Steps)	Emergence Periods (Time Steps)
Flocking	150	45,000	1-10,000 20-001-30,000 40,001-45,000	10,001-20,000 30,001-40,000
Pedestrian	382	45,000	1-10,000 20-001-30,000 40,001-45,000	10,001-20,000 30,001-40,000
Traffic	2522	65,000	1-15,000 25,001-40,000 55,001-65,000	15,001-25,000 40,001-55,000

Table 5.2: Experimentation parameters used during each simulation run

Category	Variable	Value	Description
CUSUM	h	4	The CUSUM threshold plus or minus. Outside this range, the CUSUM triggers a feedback detection event. This was selected through experimentation.
CUSUM	k	1	The significance level outside which a CUSUM observation begins to look suspicious. This corresponds to approximately 2 standard deviations.
CUSUM	M	1	How far the CUSUM window is pushed after each analysis. This means that the oldest observation is deleted.
Observation	Averaging Window	5	The number of consecutive individual values for each variable that are averaged to provide 1 value for that variable's sliding observational window.
Observation	M	10	How far the sliding observational window is pushed after each analysis. This means that the M oldest observations are deleted.
Model Selection	LASSO Window	500	The size of the sliding observational window during the initialisation phase required before model selection will be executed. This is to provide a large sample size on which to base the model selection.

Finally, Table 5.2 presents the set of parameters that remain static throughout the case study. These parameters are used during the CUSUM analysis, observation and model selection components of the DETect algorithm. While different values for each parameter may affect DETect's performance, the effect of altering these parameters is not considered during this case study. Instead, the focus is on the set of parameters that are deemed the most central and unique to DETect for both feedback and evaluation detection. This decision is discussed in more detail later in the chapter when potential threats to the validity of the study are outlined (*cf.,* Section 5.5).

5.1.2 Stage 1: Model Selection

Each multiple-linear regression (MLR) model chosen by DETect is composed of a number of internal-external variable pairs. The first stage of the study evaluates the effectiveness of this process. In chapter 4, the internal and external variables monitored in each model are outlined. These variables include internal and external variables that impact agent decision making, such as speed and heading, as well as a number of other variables that are purely random or remain static throughout each simulation. These variables are included to represent variables that may be used by an agent but are not directly related to the behaviour that generates emergence or describe their environment. These have been included to enable model selection to be evaluated. From a system model perspective, variables that describe the agent and its environment are deemed to be potentially *useful* to DETect. Variables that are entirely random or static, are deemed as potentially *non-useful*.

5.1.2.1 Evaluation Criteria

To evaluate the performance of model selection in DETect, variable pairs selected across all agents are observed. Including all variables in the model would render analysis prohibitively expensive for agents with a very high number of internal and external variables. Therefore, to be deemed useful, this model selection process should demonstrate the ability to exclude some variable pairs from the selected model so that not all possible variables available to the agent need to be monitored and analysed. Although it is necessary to reduce the number of relationships monitored, this elimination process should not be random. Instead, variables that contain no information should be excluded in greater numbers than variables that do contain some useful information. Of course, in the ideal case, all the static/random variables should be eliminated with only useful variables remaining in the model. This, to the best of our knowledge, is not possible in an automated manner. Nevertheless, the inclusion of some non-useful variables in the model is acceptable if the change detection and consensus formation stages are resilient enough to cope with their inclusion. However, a model composed entirely of variables that provide no useful information should not occur, as any inferences drawn about emergence from this model

Table 5.3: DETect feedback detection evaluation factor

Factor	**Description**	**Levels (Pedestrian & Traffic)**	**Levels (Flocking)**
Regression Window	How large is the sliding window of observations used for Regression Analysis	LOW - 20 HIGH - 40	LOW - 20 HIGH - 40
CUSUM Window	How large is the sliding window used by the CUSUM algorithm	LOW : 80 MEDIUM : 100 HIGH : 120	LOW : 10 MEDIUM : 20 HIGH : 30

during analysis will be entirely without basis.

Therefore, DETect's model selection process can be said to have succeeded, if it satisfies the following evaluation criteria:

- **MS1:** For all agents, DETect selects a subset of all possible internal-external variable pairs.
- **MS2:** The number of useful variables selected is higher than non-useful and DETect should never select only non-useful variables for any agent

5.1.3 Stage 2: Feedback Detection

The second stage of the study focusses on evaluating feedback detection, with a multi-level factor experiment design used across each model. The goal during this stage of the study is to determine the combination of parameters that achieves the best feedback detection performance for DETect. Once this is determined, the best performance achieved is evaluated to determine if it is sufficient to deem DETect's feedback detection algorithm successful.

The parameters (factor) and their levels are described in Table 5.3 with the *Regression Window* size set to one of two levels, and the *CUSUM Window* size set to one of three levels. The *CUSUM Window* sizes are consistent for both the pedestrian and traffic model, however it was necessary to use different sizes for the flocking model to achieve useful performance. This is discussed in more detail in Section 5.4. Each combination of factor level was replicated 10 times for each simulation model, with both factorial analysis of variance (ANOVA) and a one-tailed t-test performed to identify the important factors and to evaluate if DETect generates a statistical significant higher number of events during both change periods compared to non-change periods.

5.1.3.1 Evaluation Criteria

To evaluate the performance of feedback detection in DETect, the number of events generated by the *Change Detection Unit* of all agents was observed throughout each simulation. Feedback detection can be said to be effective if for some combination of parameters DETect satisfies the following evaluation criteria across all models:

- **FD1:** The number of feedback events generated by DETect across all agents is statistically significantly higher during *periods of emergence formation* compared to periods when no change is happening to the emergent system state.
- **FD2:** The number of feedback events generated by DETect across all agents is statistically significantly higher during *periods of emergence evaporation* compared to periods when no change is happening to the emergent system state.

The method used to identify these formation and evaporation periods is discussed in Section 5.2, with a more detailed description of the algorithm provided in Appendix A.

5.1.4 Stage 3: Consensus Formation

The final stage of the case study focusses on consensus formation between agents on the presence of an emergent event in the system. As with Stage 2, a multi-level factor experiment design is used across each model, to first determine the combination of parameters that achieves the best results, before ultimate performance is evaluated using the best combination.

The parameters and levels used during this stage of the study are outlined in Table 5.4, with each factor set to one of two levels. Each combination of factor level was replicated 5 times for each simulation model, giving a total of 80 replications per model. The results of the second stage are used to provide suitable levels for both the *Regression Window* and *CUSUM Window* in each model. Factorial ANOVA and a one-tailed T-test is performed to identify the important factors and to evaluate the number of emergence detection events generated by agents during *change periods* compared to non-change periods (to determine if they are statistically significantly higher). Results are considered significant when the T-test returns values of $p < 0.05$.

Table 5.4: DETect consensus formation evaluation factors

Factor	Description	Levels
Minimum Neighbourhood Size	What is the minimum number of neighbours needed before DETect will begin gossiping	LOW : 8 HIGH: 16
Maximum Neighbourhood Size	What is the maximum number of neighbours DETect will consider as potential gossiping partners	LOW: 20 HIGH: 30
Feedback Detection Memory	How long will DETect remember a recent feedback detection event	LOW: 5 HIGH: 10
Consensus Threshold	How large is Ev required to be before an emergence detection event is generated	LOW: 0.25 HIGH: 0.40

5.1.4.1 Evaluation Criteria

To evaluate the performance of consensus formation by DETect, the number of emergence detection events generated by the *Collaboration Unit* across all agents was observed throughout the simulation. As this involves all components of DETect, feedback detection and consensus formation, this stage of the evaluation concerns the overall effectiveness of the DETect algorithm. In particular, the evaluation is concerned with determining how effectively DETect performs when detecting emergence, with the combination of factors for consensus formation that best facilitates this also explored. The implicit goal of detection is to allow appropriate action to be taken to either mitigate or leverage the effects of emergence. The nature of emergence is unpredictable in advance, so it is not possible to say with certainty that action can be taken once the emergent state has become established. Therefore, detection should occur while the formation or evaporation of emergence is still taking place to allow timely action to be taken. DETect can be said to effectively detect emergence if the following performance requirements are satisfied across all models:

- **CD1:** The number of emergence detection events generated by DETect across all agents is statistically significantly higher during *periods of emergence formation* compared to periods when no change is happening to the emergent system state.
- **CD2:** The number of emergence detection events generated by DETect across all agents is statistically significantly higher during *periods of emergence evaporation*s compared to periods when no change is happening to the emergent system state.

Table 5.5: Summary of Objective Measures of Emergence

Model	Objective Measure
Flocking	The percentage of patches with no agents within a 2 radius.
Pedestrian	The number of lanes in the system
Traffic	The number of manholes reporting congestion for their associated street segment

- **CD3:** Emergence detection events are generated while the emergence formation or evaporation is taking place, to allow an interested party receiving the events to take appropriate action.

5.2 Evaluation Baselines

To facilitate evaluation, the number of events (feedback detection or emergence detection) generated across all agents is recorded every 50 time steps during each simulation run. At the same time, the objective measure of emergence for each model is monitored at the same frequency. These objective measures of emergence were described in Chapter 4, with a summary presented in Table 5.5. Each objective measure is a centralised monitor of some system variable that describes the emergent system state. In this sense, they are similar to the variable based emergence detection approaches described in Chapter 2 (*cf.,* Section 2.3) and provide a baseline against which DETect can be evaluated.

DETect's feedback and emergence detection events should coincide with emergent events occurring in the system. Emergent events are defined as the formation or evaporation of emergence e.g., the formation and dispersion of flocking behaviour in the flocking model, or congestion in the traffic model. To identify when these periods occur, the time series of the objective measure for each simulation run is processed following each simulation. A low-pass butterworth filter[1] [Parks and Burrus, 1987] is applied with a subsequent first order difference used to identify these periods of change. This processing supports the identification of periods when emergence formed and evaporated during each simulation run and, therefore, the periods when DETect should detect both feedback and emergence. For a detailed description of this processing step, please refer to

[1]A low-pass filter is a one that allows frequencies lower than a certain frequency threshold to pass and attenuates signals with a frequency higher than the cutoff frequency. Butterworth filters are a type of low-pass filter designed to have as flat a frequency response as possible for frequencies lower than the cutoff frequency. It has the effect of reducing high frequencies noise producing a smoother signal or time-series. The effect of this filtering is illustrated in greater detail in Appendix A.

Appendix A.

An illustrative example is provided in Figure 5.1 for each of the three models used in the evaluation. Each plot shows the absolute value of the objective measure (black) at intervals of 50 time steps. In addition, the associated time-series of emergence detection events generated by DETect across all models has been plotted in red. The green regions correspond to periods when emergence formed in the system and the yellow periods correspond to periods when emergence evaporated. The rest of the simulation is considered as a *Non-Change* period, including sustained periods when emergence is present in the system.

To illustrate this, consider the Flocking model from Figure 5.1. The emergent flocking behaviour forms at approximately time 200 and lasts until approximately time 400. During this period, DETect should only generate feedback detection and emergence detection events for approximately the first 60 intervals. Any events generated after this are deemed to have occurred during a *Non-Change* period as the heuristic measure has deemed the objective measure to be stable from this point. Similarly, when the flocking behaviour evaporates at time 400, the yellow period indicates the evaporation period in which DETect should generate feedback and emergence events. It is possible that for some agents, DETect may generate a detection event shortly after the conclusion of a formation or evaporation period. While it is possible that this detection event is prompted by the emergent event in the system, it is considered to have occurred during a *Non-Change* period and therefore to be a false positive. This may be pessimistic, however a strict boundary between periods was adopted during evaluation.

The formation and evaporation of emergence in both the Flocking model and the Pedestrian model are similar in that they occur at regular intervals across each simulation run. This is not the case for the Traffic model however, with the size of the map and the time it takes agents to converge meaning that emergence formation is more irregular and formation and evaporation periods can occur closer together. This is demonstrated in the illustrative example provided in Figure 5.1, where the formation of congestion can lead to a plateau in the objective measure corresponding to gridlock. Nonetheless, the same principles apply when analysing the time series of the objective measure for the model.

Once the formation and evaporation periods have been identified, the average number of events generated every 50 time steps during formation, evaporation and non-change periods are determined. For each simulation there is the initialisation period for model selection, followed

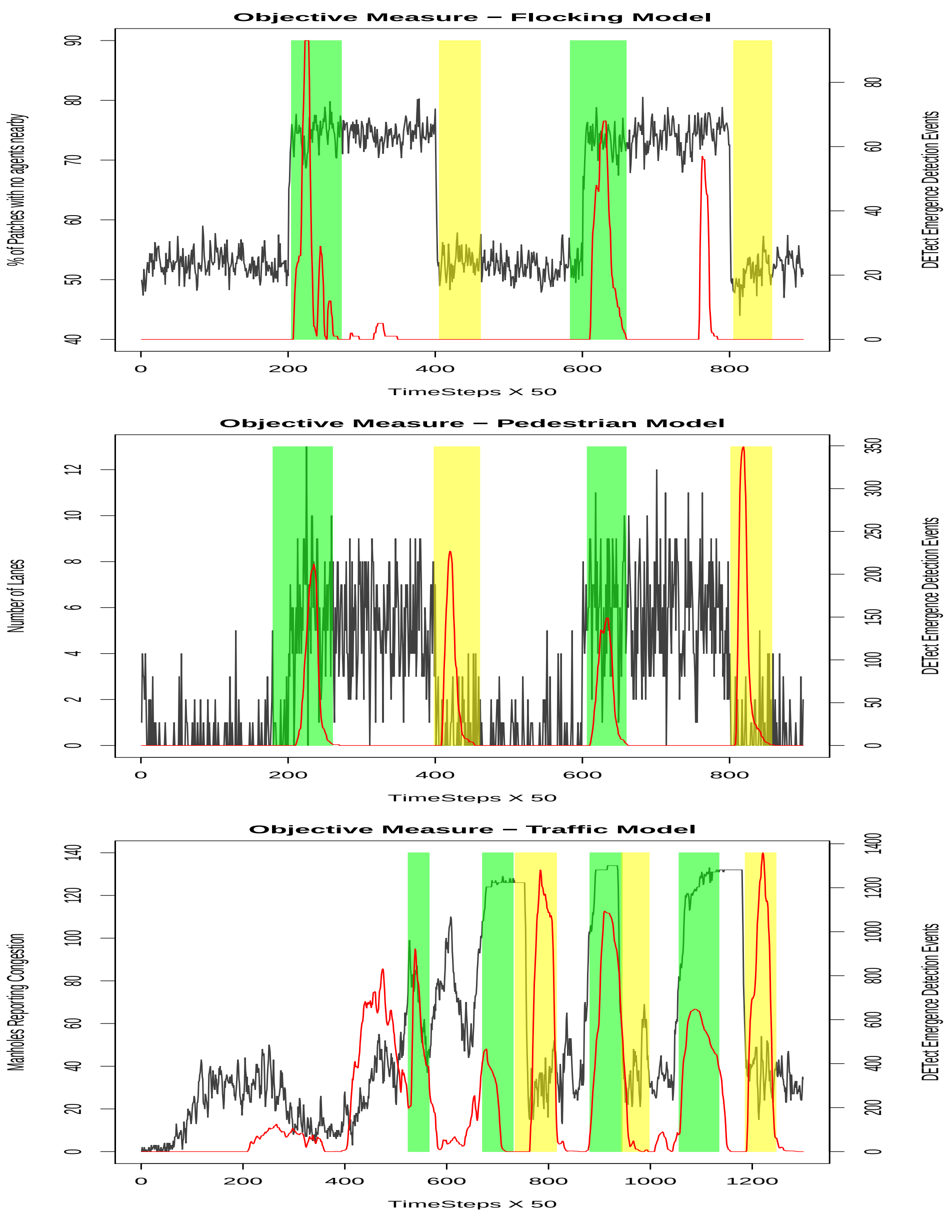

Fig. 5.1: ***Time series of objective measures:*** *Illustrative example of the time series of the objective measure (black) and DETect emergence detection events (red) from each of the simulation models with emergence formation periods (green) and evaporation periods (yellow) highlighted.*

by a seeding period where agents are compiling sufficient samples to fill the *CUSUM Windows*. Once this process is complete, DETect can begin looking for changes in the agent's environment that may signify feedback from emergence. The length of this period is a factor of the length of both the *Regression Window* and *CUSUM Window* and so differs across runs. As a result, this period has been excluded from the evaluation of each simulation run.

Finally, it should be noted that each objective measure is designed to monitor one type of pre-defined emergence in each model. It is possible that other types of unknown emergent behaviour are present in the models that are not captured by each objective measure. This possibility could lead to a high number of detection events generated by DETect being deemed false positive during the evaluation. The reason for this is that DETect is not targeted at a specific type of emergence and instead should generally detect emergence when it forms and evaporates. As a result, DETect may accurately identify emergent behaviour which is missed by the objective measure and therefore the measured accuracy of DETect in this thesis would be pessimistic.

5.3 Results and Analysis

5.3.1 Model Selection

In this section, the performance of model selection in DETect with respect to the evaluation criteria specified earlier is analysed. For each simulation model, the MLR models selected across all agents for 60 simulations were analysed, with the detailed results outlined in Table 5.6. In general the model selection process performed well with a significant number of internal-external relationship pairs excluded in all models. The most efficient performance was achieved in the Pedestrian model where DETect selected 3.14 variable pairs per agent from a possible 25.

Across all models, DETect never selected a model composed entirely of non-useful variables i.e., all selected internal and external variables are non-useful. In both the Flocking and Traffic models, DETect never selected models composed only of non-useful internal or non-useful external variables across all 60 runs. In the Pedestrian model, a small number of agents, 406 out of $22,920$ (1.77%), selected models composed exclusively of non-useful internal variables, while almost 6% selected models composed exclusively of non-useful external variables.

Although useful variables were almost always selected, DETect did include some non-useful

Table 5.6: Model Selection Results For All Models

	Flocking	**Pedestrian**	**Traffic**
No. of Agents	9,000	22,920	151,276
Total No. Variable Pairs	24	25	15
Average Selection Results			
No. of Variable Pairs Selected	7.62	3.14	8.05
Agents with only Non-useful Variables	0.00%	0.00%	0.00%
Agents with only Non-useful Internal variables	0.00%	1.77%	0.00%
Agents with only Non-useful External variables	0.00%	5.91%	0.00%
Agents with only Useful variables	1.23%	11.83%	6.21%
Agents with only Useful internal variables	24.63%	82.52%	93.60%
Agents with only Useful external variables	2.84%	14.80%	6.53%

variables in a high proportion of agents across all models. In the Flocking model, only 1.23% of agents had models composed entirely of variables deemed useful, with Pedestrian and Traffic models exhibiting slightly higher figures with 11.83% and 6.21% respectively. This was caused primarily by a failure to exclude all non-useful external variables with DETect achieving this in less than 15% of all agents in each simulation model. Nonetheless, DETect's feedback and emergence detection mechanisms proved robust to the inclusion of these variables in the selected models, as illustrated in the sections below. These results, in the context of the evaluation criteria, are discussed in Section 5.4.

5.3.2 Feedback Detection

In this section, the performance of feedback detection in DETect with respect to the evaluation objectives specified earlier is analysed. This is achieved by monitoring the number of feedback detection events generated by the *Change Detection Unit* of all agents throughout each simulation, with events being measured every 50 time steps. To improve readability, the full tables containing the analysis of variance (ANOVA) and p-values generated from Student's T-Tests are presented in Appendix B, with specific values from these tables referred to in the text below where necessary.

Table 5.7: **Flocking model - Feedback detection detailed results:** Mean number of feedback detection events generated for each factor level in the flocking model for each emergence period type.

Regression Window	**CUSUM Window**	**Period Type**		
		Non-Change	**Formation**	**Evaporation**
High	High	24.73	20.84	32.95
	Medium	7.80	10.62	6.30
	Low	4.62	7.94	3.82
Low	High	4.28	5.61	2.63
	Medium	2.50	3.52	2.00
	Low	1.53	2.75	1.16

5.3.2.1 Flocking Model

Figure 5.2 shows the average number of feedback detection events generated by DETect, during simulation runs of the flocking model for each combination of the *CUSUM Window* and *Regression Window*. The feedback detection events have been grouped based on whether emergence was forming, evaporating or not changing when the event was generated (*cf.,* Section 5.2). The *Regression Window* size was a significant factor in the performance of feedback detection with ANOVA returning a p-value of less than 0.05 for all period types. This impact is outlined in Table 5.7 where the sensitivity of the algorithm at the high level of *Regression Window* size was significantly higher for each period type across all three levels of the *CUSUM Window* size. For example, the average number of feedback detection events during formation periods for the medium CUSUM level increase from 3.52 with the low *Regression Window* size, to 10.62 with the larger window.

Variation in the *CUSUM Window* size similarly demonstrated an increasing sensitivity as the window size was increased, however the magnitude was not as significant as with the *Regression Window*. An exception to this occurred when both factors were at the higher level with the feedback detection demonstrating over-sensitive behaviour. This combination of factors is the only one that exhibited more feedback detection events occurring during evaporation periods than non-change periods. However, at this combination, the mean number of feedback events during non-change periods was 24.73, meaning 16.5% of agents were experiencing a false positive changing relationship at any moment. This proportion of agents is too high and, in addition to the poor performance during formation periods, this combination of factors was excluded.

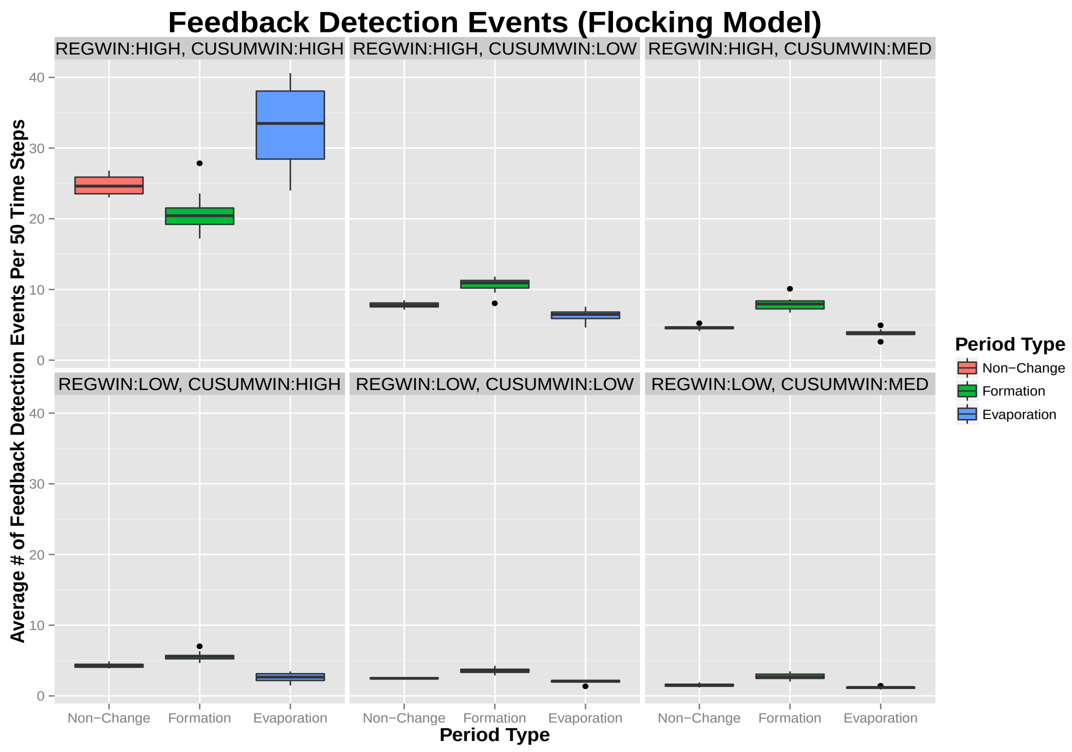

Fig. 5.2: ***Flocking model - Feedback detection results:*** *All combinations of the CUSUM Window (CUSUMWIN) and Regression Window (REGWIN) was evaluated.*

A one-tailed Student's T-test of the difference in the average number of feedback detection events during formation periods compared to non-change periods returned significant p-values (less than 0.05) when the *CUSUM Window* was at both the low and medium level, for all combinations of *Regression Window*. However, a significant difference was not seen in the average number of events generated during periods of evaporation compared to non-change. As a result, a compromise was required for the flocking model to select a combination of parameters that achieves a higher average number of feedback detection events during periods of formation compared to non-change periods. Simultaneously, it was deemed the algorithm should generate close to the same number of events during evaporation periods as non-change periods. As a result, the combination of high *Regression Window* size and medium *CUSUM Window* size was selected

Table 5.8: **Pedestrian model - Feedback detection detailed results:** Mean number of feedback detection events generated for each factor level in the pedestrian model for each emergence period type.

Regression	**CUSUM**	**Period Type**		
Window	**Window**	**Non-Change**	**Formation**	**Evaporation**
High	High	1.66	20.80	20.66
	Medium	2.05	14.58	12.28
	Low	1.69	18.23	18.74
Low	High	0.12	2.49	1.79
	Medium	0.19	1.80	1.16
	Low	0.14	2.00	1.46

for use in stage 3 of the study. The p-values obtained for all combinations of both factors are presented in Table B.4 in Appendix B.

5.3.2.2 Pedestrian Model

Results for the pedestrian model are illustrated in Figure 5.3, and demonstrate more promising results compared to the flocking model. As detailed in Table 5.8, a higher mean number of events were recorded for both formation and evaporation periods compared to non-change periods for all factor combinations. A one-tailed Student's T-tests of the difference between the number of events generated during non-change periods and both formation and evaporation returned significant p-values (less than 0.05) for all combinations of the *CUSUM Window* and *Regression Window*. The p-values obtained for all combinations of both factors are presented in Table B.8 in Appendix B. As with the Flocking model, moving from the low *Regression Window* size to the high significantly increased the sensitivity of the algorithm. This significance was reflected in the ANOVA undertaken with the *Regression Window* factor returning a p-value less than 0.05 for all three period types. However, unlike the Flocking model, the increased sensitivity did not result in false positives during non-change periods becoming unacceptably high.

Different levels of the *CUSUM Window* did not have the same scale of impact on the performance of feedback detection compared to the *Regression Window* in terms of the absolute number of events generated on average. Nonetheless, the ANOVA undertaken did indicate that different levels of the factor did have a significant impact on the number of events generated in all three period types (*cf.,* Section B.1). Interestingly, the changes in performance did not behave

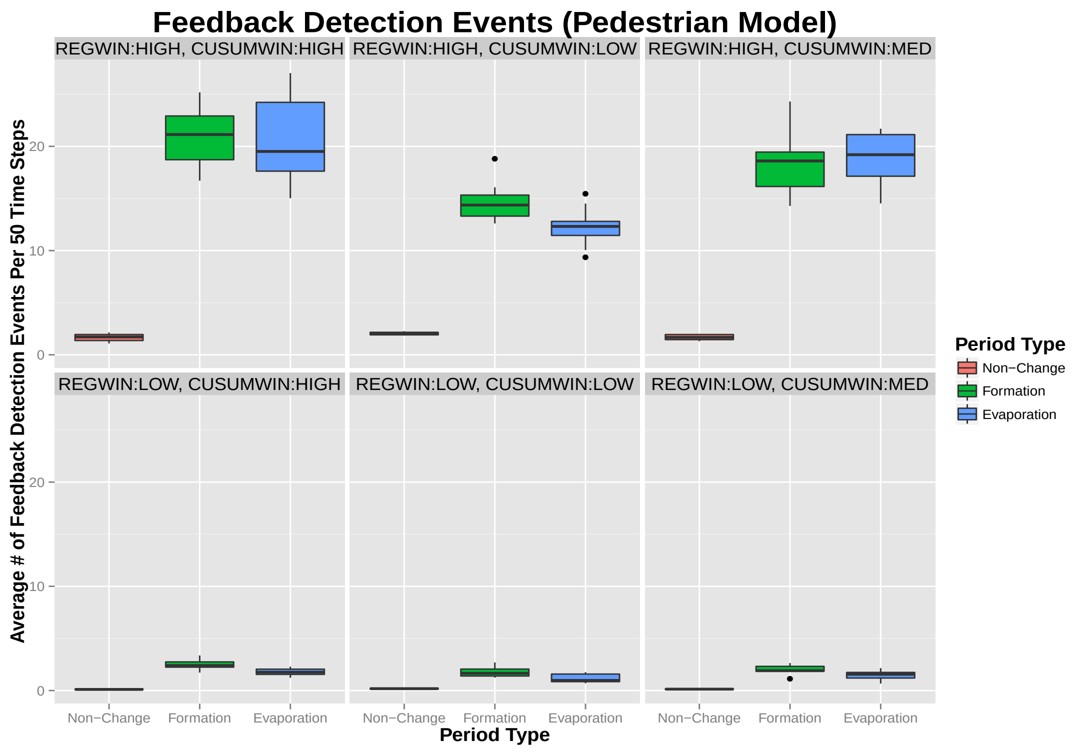

Fig. 5.3: ***Pedestrian model - Feedback detection results:*** *All combinations of the CUSUM Window (CUSUMWIN) and Regression Window (REGWIN) was evaluated.*

linearly, with a slight decrease in sensitivity from the high-level compared to the medium-level, with sensitivity increasing once more when the window size was reduced to the lower level. The best performance was achieved when both factors were at the high level with a similar average number of events generated during formation, 20.80, and evaporation periods 20.66. False positive events during non-change periods had a mean of 1.66.

5.3.2.3 Traffic Model

Figure 5.4 shows the average number of feedback detection events generated during the traffic model for each factor combination. As with the other models in this study, the size of the *Regression Window* significantly affected the sensitivity of the algorithm, while changes in the

Table 5.9: **Traffic model - Feedback detection detailed results:** Mean number of feedback detection events generated for each factor level in the traffic model for each emergence period type.

Regression Window	CUSUM Window	Period Type		
		Non-Change	Formation	Evaporation
High	High	110.44	221.65	200.29
	Medium	122.85	170.33	172.54
	Low	120.00	186.10	136.18
Low	High	7.46	12.80	15.12
	Medium	8.83	11.28	10.70
	Low	7.87	11.28	1.46

CUSUM Window size had a smaller impact on performance. Table 5.9 demonstrates this, with fewer than 20 feedback detection events being generated during any period when the *Regression Window* size was set to the lower level. Conversely, at the higher level, the average exceeded 100 events for all periods, and at the higher level of the CUSUM window, exceeded 200 events for both emergence formation and evaporation. As the model is composed of 2500 agents, an average of less than 20 events during transition periods is too low and so the low level of the *Regression Window* is deemed to be too insensitive for use.

The algorithms performed acceptably at the higher regression window size with a significantly higher number of events being generated during formation and evaporation periods, compared to non-change periods. The difference between these periods was not as stark as in the Pedestrian model, with a large number of false positives being generated, however the difference between change periods and non-change periods was deemed sufficiently large. This is especially the case for the combination of high *Regression Window* and high *CUSUM Window* where the average number of events generated during formation periods was over twice that of non-change periods. Evaporation periods displayed a similar high number of average event regularity at this parameter level also. This is confirmed by a one-tailed Student's T-Test comparing both change periods to non-change periods with a significantly higher average number of feedback events generated when both parameters were set to *High*. The p-values obtained for all combinations of both factors are presented in Table B.12 in Appendix B. As a result, the High level for both the *Regression Window* and *CUSUM Window* were selected to use during the third stage of the study. The selection of suitable values for each of these parameters is discussed further in Section 5.4.

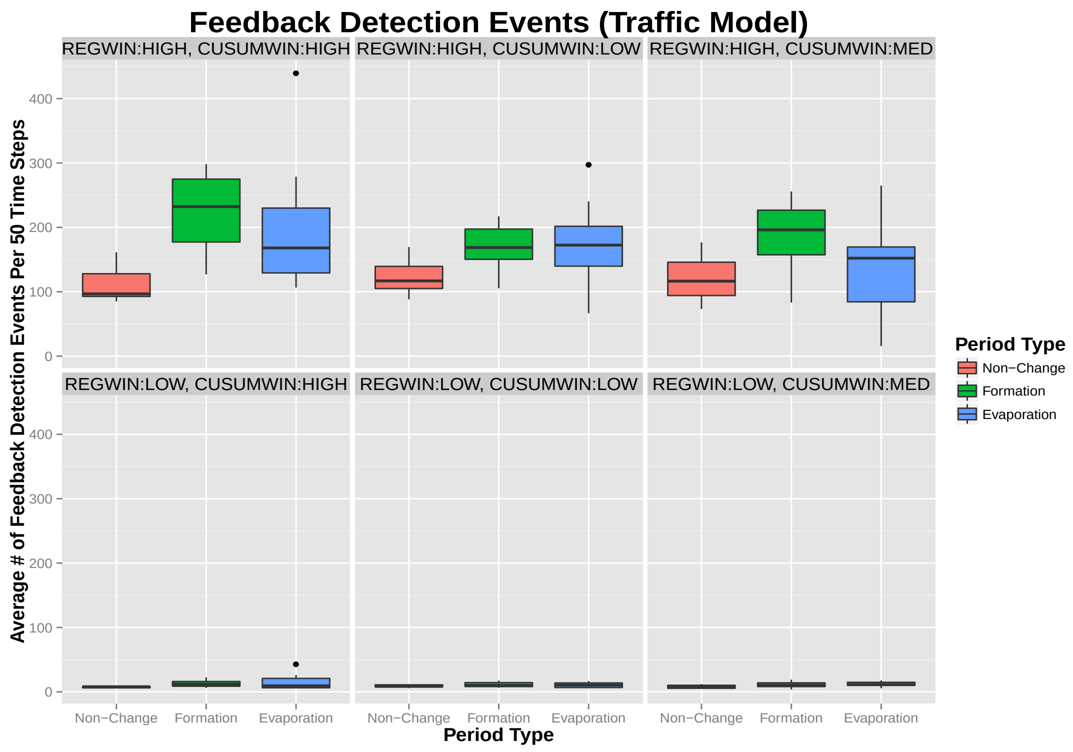

Fig. 5.4: ***Traffic model - Feedback detection results:*** *All combinations of the CUSUM Window (CUSUMWIN) and Regression Window (REGWIN) were evaluated.*

5.3.3 Consensus Formation

In this section, the performance of DETect when forming consensus on the presence of emergence with respect to the evaluation objectives specified earlier is analysed. As with feedback detection, this is achieved by monitoring the number of detection events generated throughout each simulation, with events being measured every 50 time steps. The consensus protocol involves 4 parameters, *Consensus Threshold, Feedback Detection Memory length, Neighbourhood Size Minimum and Neighbourhood Size Maximum*, each with two levels. The analysis for each model is structured to first examine the impact of the Consensus threshold and memory length parameters. Next, the two neighbourhood parameters are examined together. Finally, the best performance achieved is analysed with an emphasis placed on the timeliness of the detection

Table 5.10: Flocking model - Detailed consensus threshold and memory length results: Mean number of emergence detection events generated for each factor level in the flocking model for each combination of consensus threshold and feedback detection memory length factors

Threshold Level	Memory Length	Period Type		
		Non-Change	**Formation**	**Evaporation**
High	High	14.25	67.76	12.48
	Low	2.31	14.76	0.64
Low	High	56.61	101.62	68.47
	Low	12.93	42.98	10.15

events. The full analysis of variance (ANOVA) using all 4 factor may help domain experts when applying DETect on other future case studies. However, this is not material to the evaluation criteria outlined in Section 5.1.4 and is therefore supplied, along with the results of Student's T-Tests, in Appendix C for completeness.

5.3.3.1 Flocking Model

Consensus Threshold And Memory Length

Figure 5.5 illustrates the results of consensus formation for each combination of *Consensus threshold* and *Feedback Detection Memory length* across all levels of neighbourhood size factors. These results are outlined in detail in Table 5.10 and demonstrate that the length of time each agent remembers a feedback detection event significantly impacts DETect's performance when forming consensus on the existence of emergence. In particular, a high memory length leads to a larger increase in the number of emergence detection events generated during non-change and evaporation periods compared to formation periods.

This is most evident at the low level for *Consensus Threshold* level, where detection rates increase approximately 5 and 7 times respectively for non-change and evaporation periods, compared to approximately 2.5 during formation periods. Moving from the higher to the lower *Consensus Threshold* level has a similarly disproportionate impact on the number of emergence detection events generated during non-change and evaporation periods compared to formation periods. However, the change in sensitivity is less significant than for feedback detection memory length.

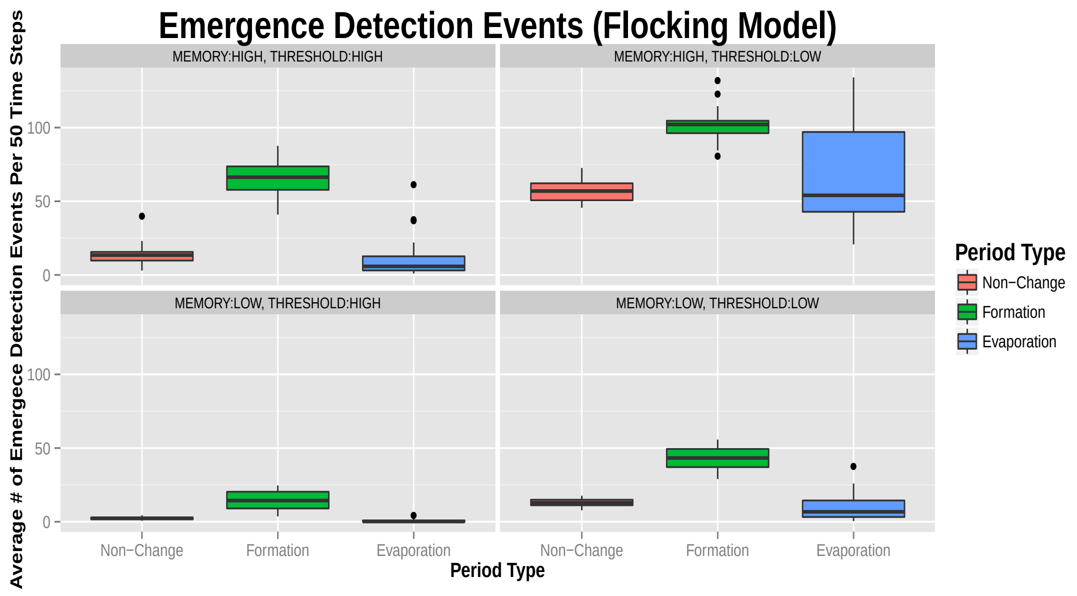

Fig. 5.5: ***Flocking model - Consensus formation and emergence detection results:*** *Results for all combinations of consensus threshold and feedback detection memory length parameters are illustrated.*

Neighbourhood Factors

The impact of the neighbourhood size factors is described in Table 5.11. These results demonstrate that performance does not markedly change between levels of these factors, with the mean number of emergence detection events generated for both non-change periods and formation periods remaining consistent throughout. However, performance during evaporation periods is impacted by the minimum neighbourhood size factor, with more than twice the number of events being generated at the higher level, as well as an increased variability in the number of events generated across each run.

Best Performance

A "High" setting for both consensus threshold and feedback detection memory length achieved

Table 5.11: **Flocking model: Neighbourhood parameter detailed results:** Average number of emergence detection events generated in the flocking model for each combination of Neighbourhood Minimum and Maximum Size.

Neighbourhood Minimum Size	**Neighbourhood Maximum Size**	**Period Type**		
		Non-Change	**Formation**	**Evaporation**
High	High	23.30	59.10	35.99
	Low	21.10	53.96	27.57
Low	High	20.91	52.71	13.16
	Low	20.80	58.35	14.96

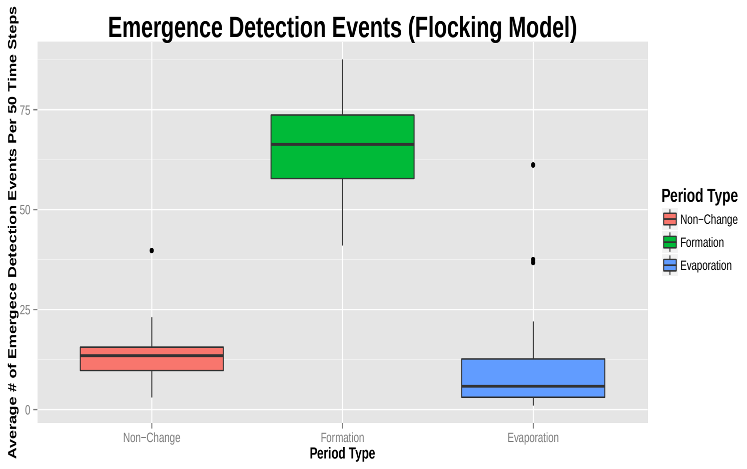

Fig. 5.6: ***Flocking model- Best results:*** *A combination of HIGH Consensus Threshold and HIGH Feedback Detection Memory Length achieved the best results in the flocking model.*

the best performance in the flocking model across all levels of neighbourhood size factors. This is illustrated in Figure 5.6 with a significantly higher number of events detected on average during emergence formation periods, 67.76, compared to non-change periods, 14.25. However, the average number of emergence detection events during evaporation periods is not statistically higher than during non-change periods with only 12.48 events generated on average per time interval. This is a legacy of the poor performance of the feedback detection mechanism that was encountered during flocking evaporation periods in stage 2 of the study.

Figure 5.7 further illustrates this by overlaying all formation periods and evaporation periods across all runs with this factor combination to create an average transition period for each period

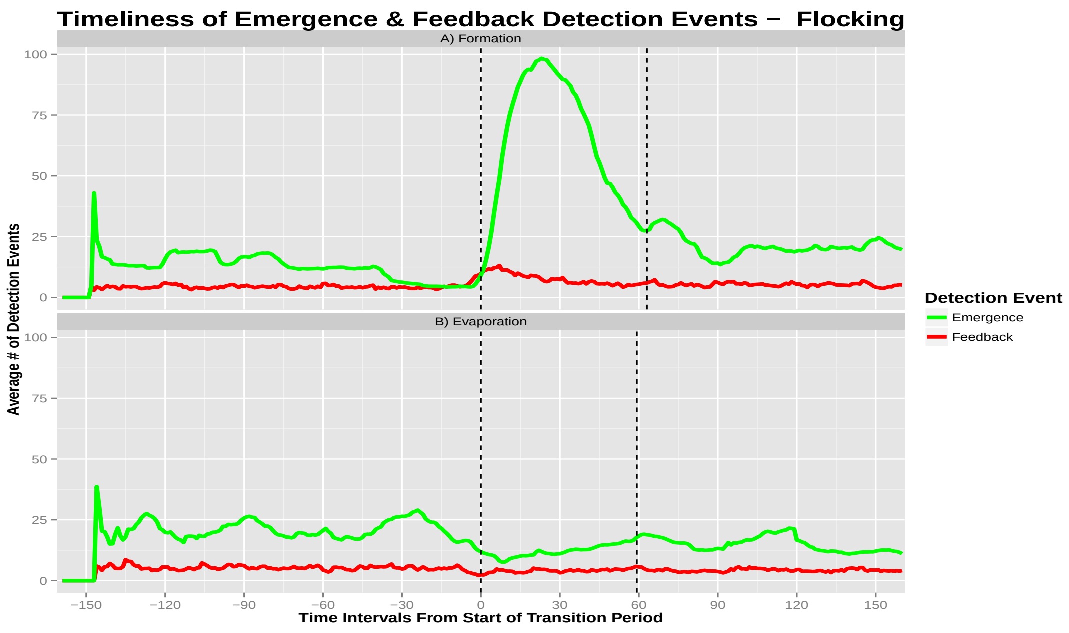

Fig. 5.7: ***Flocking model - Timeliness of detection:*** *Timeliness of emergence detection is considered in relation to the start and end of each transition period (dashed lines).*

type. The vertical dashed line at time step 0 indicates the start of the transition period with the second vertical line corresponding to the end of the transition period, based on the average length of such periods across all runs. The average number of both feedback and emergence detection events generated every interval (50 time steps) are plotted in relation to the start of each transition period. The period to left of the starting line corresponds to time before the transition period begun. Events here are generated during non-change periods that are either at the start of the simulation or after the previous transition period in the simulation finished. To the right of the second vertical line corresponds to time after the average transition period has finished. Detection events that occur during this non-change period are deemed "late", occurring after the transition period has finished and before the next transition period commences.

Formation periods during flocking last for 63 time intervals on average. They result in a slight increase in the average number of feedback detection events generated by the agents (red line). Each agent that generates a feedback detection event remembers it for 10 subsequent intervals. This, coupled with the high consensus threshold of 0.40, results in a significant increase in the number of agents generating emergence detection events (green line). A maximum of 98 emergence detection events on average are generated simultaneously 23 intervals from the start of the formation period. This corresponds to a peak of 65% of all agents and it arrives approximately 37% into the average transition period length. As a result, an interested party receiving these events, such as an adaptation manager, would have over 60% of the transition period to react appropriately following this peak, before the emergent state becomes static again. In practise, the interested party may not know how long the transition period will last and durations are likely to differ across domains and, even, from one transition period to another. However, this illustrates the timeliness of DETect's emergence detection demonstrating that events are generated during the transition period. Finally, a noticeable spike in emergence detection events occurs at approximately time interval -150 for both formation and evaporation periods. These are "late" detection events that are caused by the previous transition period in the simulation but are generated after that transition period was deemed to have finished.

5.3.3.2 Pedestrian Model

Consensus Threshold And Memory Length

The performance of DETect for consensus formation and emergence detection in the Pedestrian model is illustrated in Figure 5.8 for each combination of the feedback detection memory length and consensus threshold. A detailed breakdown of these results is provided in Table 5.12. DETect's performance in this model builds on the strong results from the feedback detection phase with a very small emergence detection event rate being recorded during non-change periods at all factor levels. The best performance is experienced when the *Consensus Threshold* is low and *Memory length* is high, with 129.33 and 131.37 events being generated during formation and evaporation periods respectively. This is significantly higher than other factor combinations, indicating that both factors impact on performance.

Neighbourhood Factors

The performance of DETect for each combination of the neighbourhood size factors is detailed in Table 5.13. These results indicate that DETect is not significantly affected by the size of the neighbourhood parameters used during the pedestrian model simulations, with a relatively small increase in sensitivity achieved with the high minimum neighbourhood size compared to the low.

Table 5.12: **Pedestrian model - Detailed consensus threshold and memory length results:** Mean number of emergence detection events generated for each factor level in the pedestrian model for each combination of consensus threshold and feedback detection memory length factors

Threshold Level	**Memory Length**	**Period Type**		
		Non-Change	**Formation**	**Evaporation**
High	High	0.06	11.05	39.95
	Low	0.01	0.66	2.43
Low	High	0.39	129.33	131.37
	Low	0.08	39.53	61.44

This increase in the average number of emergence detection events is achieved without a similar increase in the number of false positives generated during non-change periods.

Table 5.13: **Pedestrian model: Neighbourhood parameter detailed results:** Average number of emergence detection events generated in the pedestrian model for each combination of Neighbourhood Minimum and Maximum Size.

Neighbourhood Minimum Size	**Neighbourhood Maximum Size**	**Period Type**		
		Non-Change	**Formation**	**Evaporation**
High	High	0.10	49.59	63.94
	Low	0.17	48.27	63.17
Low	High	0.11	39.17	52.45
	Low	0.15	43.52	55.62

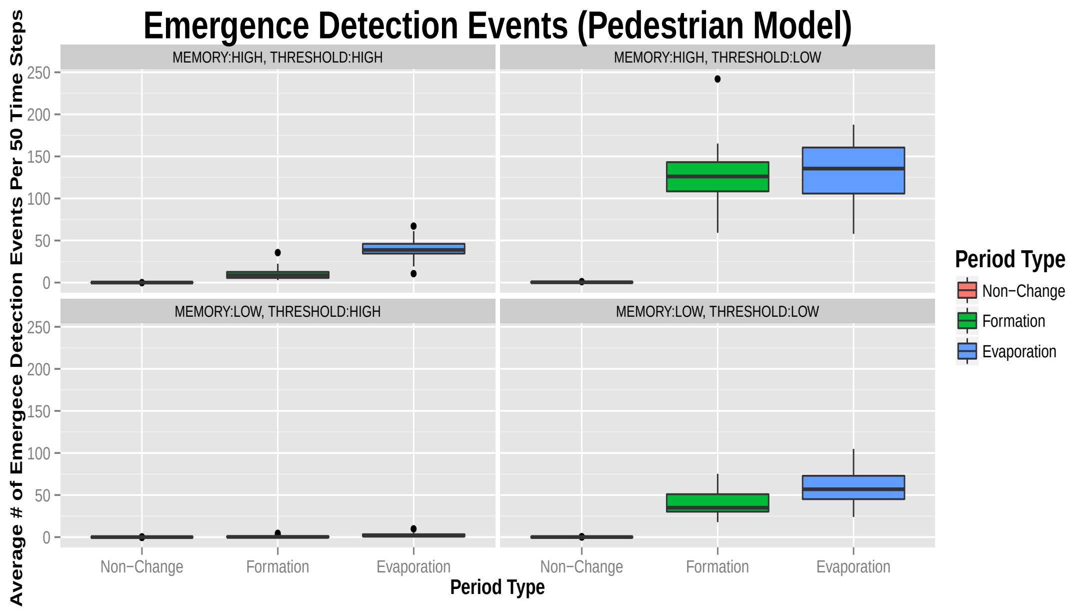

Fig. 5.8: ***Pedestrian model - Consensus formation and emergence detection results:*** *Results for all combinations of consensus threshold and feedback detection memory length parameters are illustrated.*

Best Performance

The performance of DETect for consensus formation and emergence detection in the Pedestrian model is illustrated in Figure 5.9, with a low *Consensus threshold* and high *feedback detection memory length* yielding the best results across all combinations of neighbourhood parameters. Unlike flocking, DETect generated a significantly higher number of emergence detection events during both emergence formation, 129.33, and evaporation 131.37, compared to non-change periods, 0.39.

The timeliness of these events in relation to both types of transition periods is illustrated in Figure 5.10, with an average formation period lasting 61 intervals and average evaporation period lasting 57. As with the formation periods in the flocking model, transition periods in

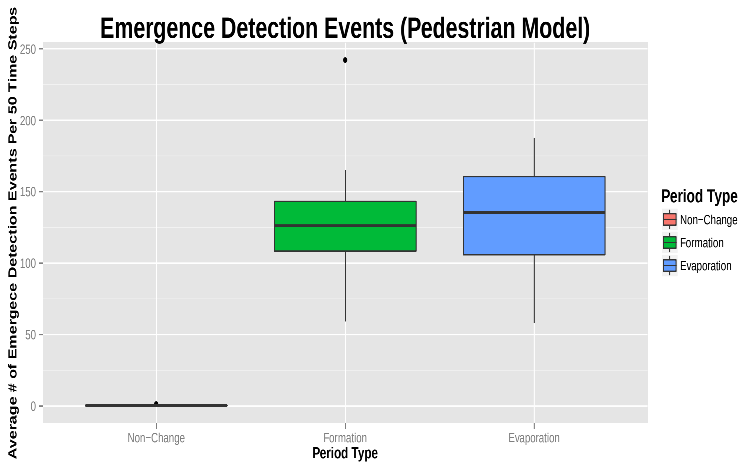

Fig. 5.9: ***Pedestrian model - Best results:*** *A combination of a LOW Consensus Threshold and HIGH Feedback Detection Memory Length achieved the best results in the pedestrian model.*

the pedestrian model correspond with an increase in the number of feedback detection events generated. This results in a significant increase in the number of emergence detection events generated across all agents.

During formation periods, a maximum of 261 emergence detection events are generated on average 30 intervals from the start of the transition period. This corresponds to a peak of 68% of all agents and arrives at the half way point of the transition period. Evaporation periods exhibit a lower maximum of 224 simultaneous emergence detection events, however this peak occurs after only 23 intervals from the start of the transition period, allowing a larger proportion of the transition period to be used to react. In both instances, an adaptation manager would receive almost all detection events associated with the formation and evaporation periods while the transition to the new emergent state was still occurring. This would theoretically allow action to be taken to mitigate or leverage the effects of the emergence by agents across the system.

5.3.3.3 Traffic Model

Consensus Threshold And Memory Length

The results for consensus formation and emergence detection in the Traffic model are illustrated in Figure 5.11, for each combination of consensus threshold and memory length, with a detailed

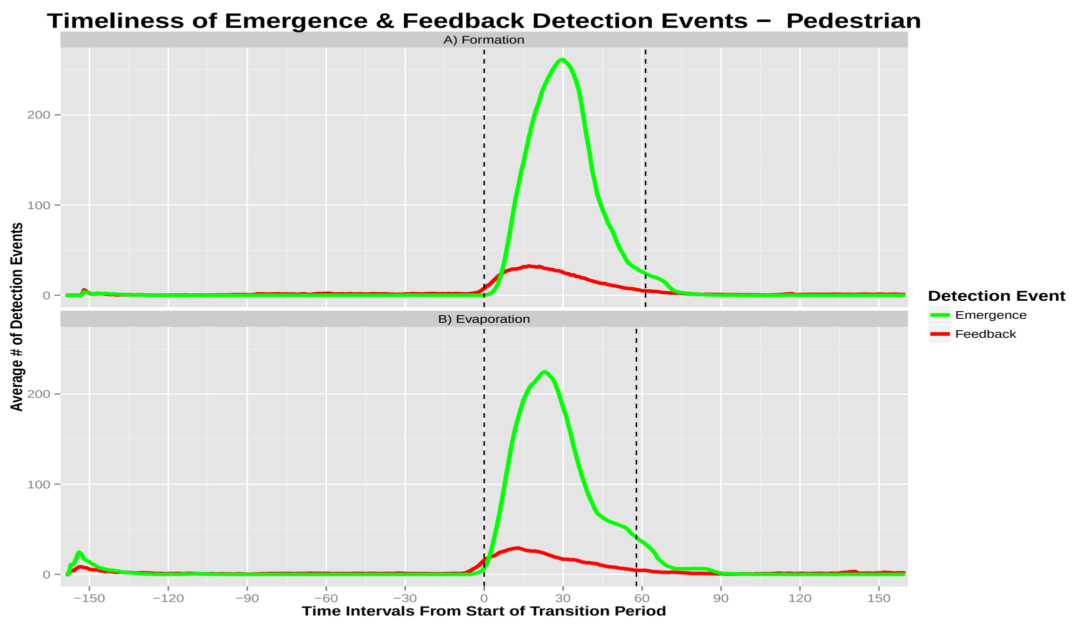

Fig. 5.10: ***Pedestrian model - Timeliness of detection:*** *Timeliness of emergence detection is considered in relation to the start and end of each transition period (dashed lines).*

breakdown provided in Table 5.14. Similar to the Pedestrian model, DETect generates a statistically significantly higher number of emergence detection events during emergence formation and evaporation periods compared to non-change periods for each combination.

Consensus threshold level has a large impact on the number of detection events observed, with the number of events more than doubling during non-change periods at the lower level. The lower level of this factor has a similar impact during both formation and evaporation periods at the lower level of the feedback detection memory length factor. However, when the *Memory length* is set to the high level the increased number of false positive during non-change periods does not see a comparative increase in detection rates during formation and evaporation periods.

Neighbourhood Factors

A detailed breakdown of the performance of DETect at each combination of the neighbourhood size factors is provided in Table 5.15. These results illustrate that the performance of DETect is not significantly impacted by these parameters, with similar results achieved for all 3 periods types for each combination of factors.

Table 5.14: **Traffic model - Detailed consensus threshold and memory length results:** Mean number of emergence detection events generated for each factor level in the traffic model for each combination of consensus threshold and feedback detection memory length factors

Threshold	**Memory**	**Period Type**		
Level	**Length**	**Non-Change**	**Formation**	**Evaporation**
High	High	152.99	424.19	407.53
	Low	72.79	222.15	207.58
Low	High	328.66	592.23	483.02
	Low	162.21	416.81	365.26

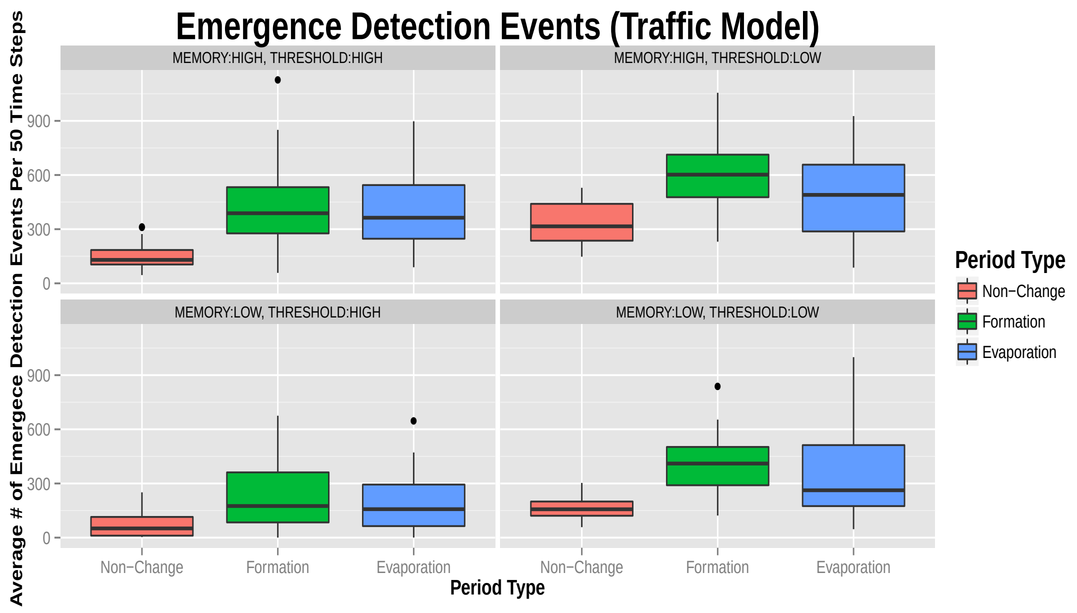

Fig. 5.11: ***Traffic model - Consensus formation and emergence detection results:*** *The results for all combination of consensus threshold and feedback detection memory length parameters are illustrated.*

Best Performance

The best achieved results for emergence detection in the traffic model are illustrated in Figure 5.12, with a high level used for both *Consensus threshold* and *feedback detection memory length*

Table 5.15: **Traffic model: Neighbourhood parameter detailed results:** Average number of emergence detection events generated in the traffic model for each combination of Neighbourhood Minimum and Maximum Size.

Neighbourhood Minimum Size	Neighbourhood Maximum Size	Period Type		
		Non-Change	Formation	Evaporation
High	High	153.51	394.76	350.59
	Low	155.10	383.66	371.05
Low	High	211.04	530.89	319.34
	Low	196.98	346.05	422.44

across all combinations of neighbourhood parameters, to achieve the these results. Similar to the pedestrian model, DETect generates a statistically significantly higher number of emergence detection events during emergence formation, 424.19, and evaporation 407.53, periods compared to non-change periods, 152.99. The timeliness of consensus formation in the traffic model does not match the performance achieved in the pedestrian model however, as illustrated in Figure 5.13.

The average formation period lasts for 76 time intervals and it results in a gradual increase in the number of both feedback and emergence detection events being generated. This results in a maximum of 504 simultaneous emergence detection events, 20% of all agents, occurring 82% into the average formation period. The rate that events are generated remains high and results in a significant number occurring after the transition period has ended. As a result, these events are counted as having occurred during a *non-change period*.

By comparison, the average evaporation period is shorter, lasting for 55 time intervals. A maximum of 450 simultaneous emergence detection events, 18% of all agents, occur 45% into these periods on average. This allows a larger proportion of the transition period to react to the changing emergent state however, as with the formation periods, the number of events remains high throughout the period with some occurring after the transition phase is deemed to have ended. The relative high number of emergence detection events that are generated at the start of this plot (time interval -160 to -100) are the result of those events that were prompted by the previous formation periods but did not occur until after the formation period was deemed

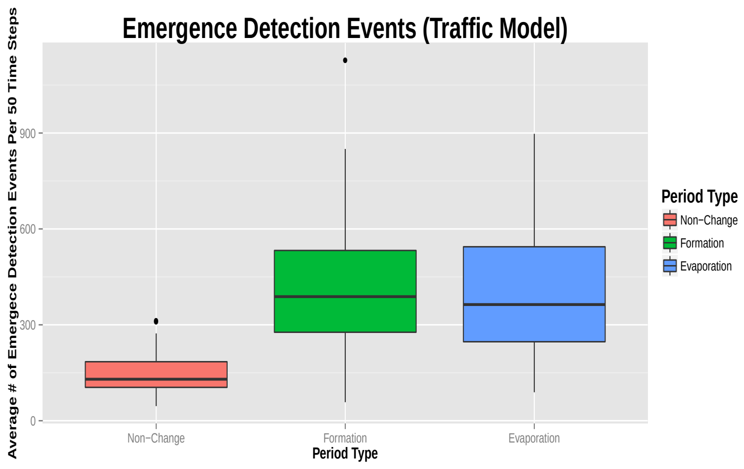

Fig. 5.12: ***Traffic model - Best results:*** *A combination of a HIGH Consensus Threshold and HIGH Feedback Detection Memory Length achieved the best results in the Traffic model.*

to have ended.

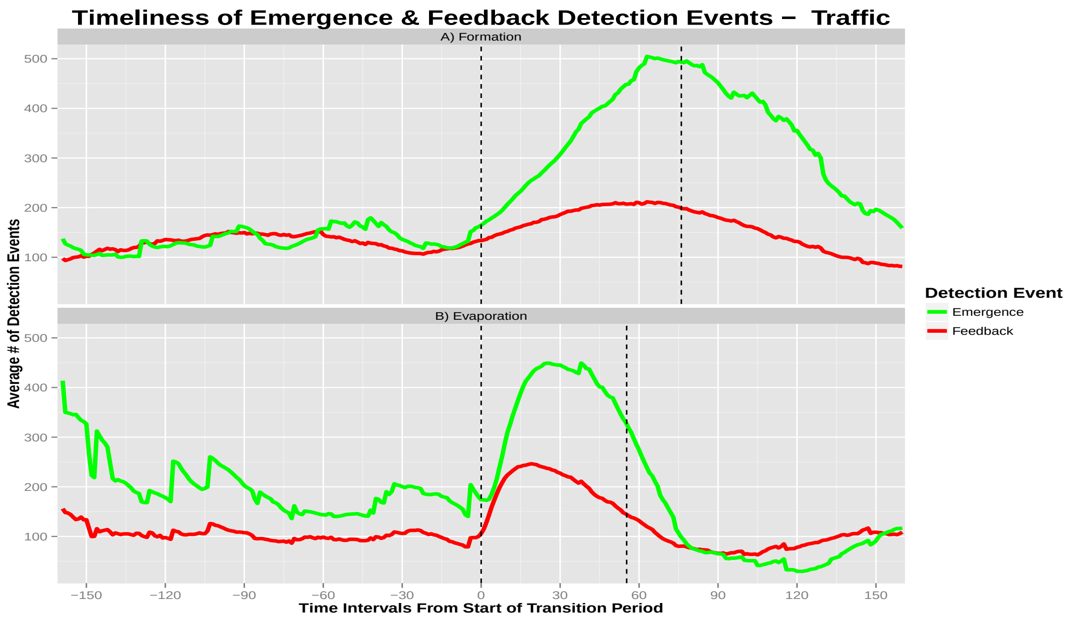

Fig. 5.13: ***Traffic model - Timeliness of detection:*** *Timeliness of emergence detection is considered in relation to the start and end of each transition period (dashed lines).*

5.4 Discussion

In this section, the performance of DETect in each stage of the case study is discussed. This performance is summarized in Table 5.16 using the evaluation criteria identified earlier (*cf.,* Section 5.1).

5.4.1 Model Selection

Evaluation criteria $MS1$ and $MS2$ concern the model selection process performed by the *Modelling Unit*. Criteria $MS1$ and $MS2$ are satisfied for each of the three models with a significant number of potential internal-external variable pairs being excluded from the final MLR selected by DETect. Additionally, in both the flocking and traffic models, the selected model is never composed entirely of either non-useful internal variables or non-useful external variables, satisfying the second component of criterion $MS2$.

In the pedestrian model, for a small number of agents, 1.77% and 5.5%, DETect does select models with exclusively non-useful internal or external variables respectively. Moreover, for all models, the number of agents who ended up with non-useful variables included in their MLR was high. However, this should be viewed in light of the overall performance of DETect at both feedback detection and emergence detection. The inclusion of these non-useful variables did not adversely affect the performance, especially in the pedestrian and traffic models, demonstrating that DETect's feedback detection algorithm is robust to such inaccuracies. It is also possible that including some mis-information in the system model may help to improve performance [Nallur et al., 2015], however this requires some additional experimentation before any definitive conclusion can be reached. Finally, these results were achieved in agents with a relatively small number of internal and external variables which potentially limits the conclusions that can be drawn, especially in the context of agents with significantly more variables. This is discussed further in Section 5.5 below.

5.4.2 Feedback Detection

Evaluation criteria $FD1$ and $FD2$ concern feedback detection and are satisfied in both the pedestrian and traffic model, with a combination of parameters identified that achieved higher detection rates during transition periods. In particular, for both models, DETect's feedback de-

Table 5.16: Summary of results against evaluation criteria

Criterion	Description	Flocking Model	Pedestrian Model	Traffic Model
MS1	All agents select a subset of all possible internal-external variable pairs.	Satisfied	Satisfied	Satisfied
MS2	The number of useful variables selected is higher than non-useful and DETect should never select only non-useful variables for any agent	Satisfied	Partially Satisfied	Satisfied
FD1	The number of feedback events generated by DETect across all agents is statistically significantly higher during periods of emergence formation compared to periods when no change is happening to the emergent system state.	Satisfied	Satisfied	Satisfied
FD2	The number of feedback events generated by DETect across all agents is statistically significantly higher during periods of emergence evaporation compared to periods when no change is happening to the emergent system state.	Not Satisfied	Satisfied	Satisfied
CF1	The number of emergence detection events generated by DETect across all agents is statistically significantly higher during periods of emergence formation compared to periods when no change is happening to the emergent system state.	Satisfied	Satisfied	Satisfied
CF2	The number of emergence detection events generated by DETect across all agents is statistically significantly higher during periods of emergence evaporation compared to periods when no change is happening to the emergent system state.	Not Satisfied	Satisfied	Satisfied
CF3	Emergence detection events should be generated while the emergence formation or evaporation is taking place, to allow an interested party receiving the events to take appropriate action.	Partially Satisfied	Satisfied	Partially Satisfied

tection performed better with a combination of a higher *Regression Window* and higher *CUSUM Window*. These factors represent the memory of each agent, and how long a period of time should be used to judge recent observations against. This suggests that the longer agent's memory is about its experiences, the better DETect will perform in detecting feedback from emergence. However, a larger memory requires a longer period of time to fill the sliding windows and therefore a longer time before DETect's initialisation phase finishes. The models examined in this study involved a static set of variables with variables neither being added or removed once the system is initialised. As a result, the length of the initialisation period was not a concern. However, this aspect of DETect's performance suggests systems that require such on-line resets may adversely affect the utility of DETect.

In the flocking model, feedback detection was effective during formation periods, satisfying evaluation criteria $FD1$. However, DETect was not effective at detecting feedback during evaporation periods in this model, failing to satisfy objective $FD2$. The limitation of DETect during emergence evaporation in the flocking model means that additional work is required to improve performance in this area. This performance aspect is in addition to the requirement for a significantly smaller series of *CUSUM Window* sizes when using the flocking model, 10-30, compared to both the pedestrian and traffic models, where consistent levels were applied across all factors, 80-120. Attempts were made to find a range of CUSUM window sizes that would perform sufficiently across all models. However, the sensitivity of the feedback detection was significantly impacted when the CUSUM window size was increased in the flocking model, with a reduction in window size creating the same effect in the Pedestrian and Traffic model. As a result, no compromise range could be found and the study focussed on the ranges that resulted in useful performance for each model. It is probable that the reason for this discrepancy is due to the nature of the emergent phenomena in each of the models. For example, the movement of the boid agents in the flocking model appears to be more dynamic compared to the traffic and pedestrian models, where collisions are forbidden, resulting in a smaller memory of the past being more beneficial.

An initial attempt to characterise this is illustrated in Figure 5.14, with a comparison of the rate of change in the mean neighbourhood size of agents throughout a simulation of each model. Neighbourhood size is defined as the number of one-hop neighbours each agent has. The positive spikes correspond to periods of emergence formation and the negative spikes correspond to periods of evaporation. Flocking demonstrates significantly larger change rates during these transition periods compared to the other models and a more dynamic environment when emergence is present compared to periods without emergence. Further experimentation is required to fully understand this variation across a large number of different systems (*cf.,* Section 5.5).

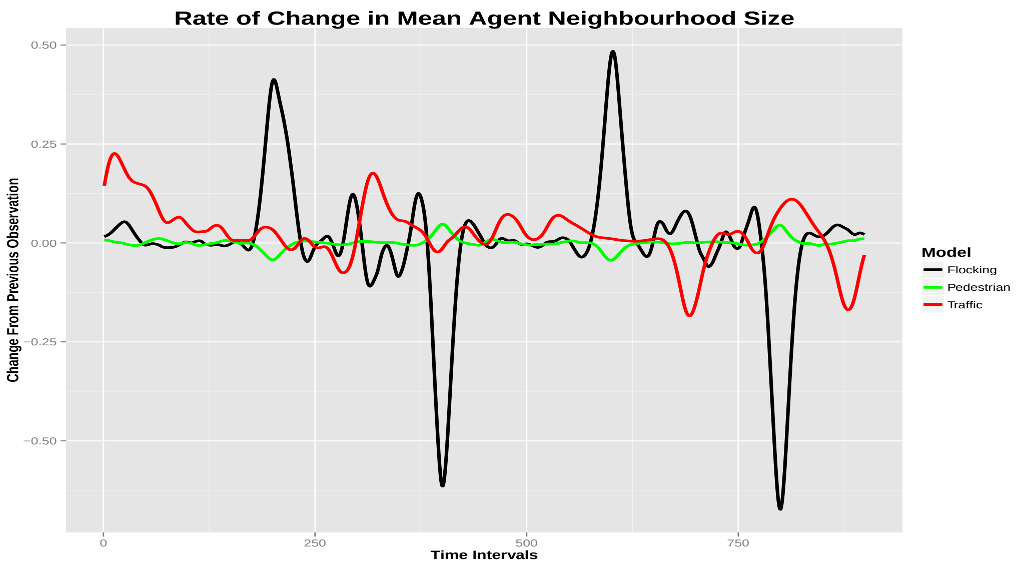

Fig. 5.14: ***Neighbourhood size rate of change:*** *A comparison of the change in the average neighbourhood size of agents during runs in each model. The flocking model demonstrated greater variation during periods of emergence formation and evaporation.*

5.4.3 Consensus Formation

Finally, evaluation criteria $CF1$ to $CF3$ concern consensus formation on the existence of an emergent event. The factor combinations that provided the best performance differed between each of the models. However, for both the pedestrian and traffic models, a combination of factors existed that provided a significantly higher number of detection events during periods of emergence formation and evaporation compared to non-change periods. Criteria $CF1$ and $CF2$ are, therefore, satisfied for both models. In the Flocking model, consensus formation performed well during periods of emergence formation satisfying evaluation criterion $CF1$. However, the performance of feedback detection during periods of emergence evaporation was not sufficiently high to satisfy evaluation criterion $FD2$, with this limitation subsequently impacting the performance of consensus formation. As a result evaluation criterion $CF2$ was unsatisfied for the flocking model.

The timing of the feedback and emergence detection events across all models coincided with the start of the formation and evaporation periods. In the pedestrian model, these periods saw a significant spike in the number of generated events with over 60% of agents simultaneously generating emergence detection events during the transition periods. This increased event rate could indicate to an adaptation manager that an emergent event is taking place, allowing appropriate steps to be taken to either mitigate or leverage the event, satisfying criterion $CF3$. A similar performance was achieved during formation periods in flocking however, as DETect did not effectively detect evaporation periods, criterion $CF3$ is, therefore, only partially satisfied in the flocking model.

In comparison, the rate of detection events in the traffic model increased gradually, with a significant number occurring after the transition period was deemed to be finished. The effect of this "lateness" is that though these detection events are prompted by the transition period, they are classified in the study as false-positives, as they occurred during non-change periods. As a result, the evaluated accuracy of DETect in the traffic model is perhaps somewhat pessimistic. Nonetheless, criterion $CF3$ is therefore only partially satisfied for the traffic model, with additional work required to improve the sensitivity of DETect in these transition periods. It is hoped that improved sensitivity in model will result in detection events that are generated in a timely manner, allowing appropriate adaptation to be undertaken before the emergent system state becomes established.

Finally, the objective measure of emergence for each of the three models is designed to monitor only one type of expected emergence (*cf.* Section 5.2). Despite this, it is possible that other, possibly multiple, types of emergents are present in each of the models but are unknown and not captured by each objective measure. In contrast, DETect is theoretically capable of operating in scenarios where two or more types of emergence behaviour are present in the system, detecting both types when they form and evaporate. This capability is not explicitly evaluated in this case study, however its potential allows for the possibility that detection events deemed to occur during *non-change periods* are actually DETect identifying another, not-predefined, emergent property that is not captured by the objective measure. In such an eventuality, the accuracy of DETect outlined in this chapter may be pessimistic as the number of false-positive detections would be artificially high.

5.5 Threats to Validity

The following aspects of the experimental set-up potentially limit the conclusions that can be drawn from the case study.

Model Types: The results demonstrate that DETect performs relatively well across each model. However, any claims regarding the general applicability of DETect for use in CAS are tempered by the nature of the models used during the experimentation. In particular, all three models involved agents that moved spatially, resulting in similar variables being used to describe both the agents and their environments across each model. Additionally, although the experimentation used models of varying scales, DETect has not been tested on truly ultra-large scale systems involving tens or hundreds of thousands of agents, or agents with thousands of internal and external variables to select from.

This represents a potential threat to the validity of both the model selection process used by DETect and also the type of emergence that DETect is capable of detecting. The use of these models was necessitated by a lack of diverse open source simulation models that exhibit emergence, where the presence of emergence is easily identifiable and, therefore, uncontroversial. For instance, almost all comparative papers that discuss emergence detection use flocking as their standard test case. Thus, the claims in this thesis regarding the effectiveness of DETect for detecting emergence are limited to similar models, where emergence is generated by agent interactions through movement.

No adaptation: The agents in the experiments adapt their behaviour relative to one another and their environment. They do not adapt in the sense of changing their inherent behavioural capabilities. This limitation was not considered significant as the focus in this thesis was on evaluating DETect's ability to detect emergence. Although agents themselves may or may not be adaptive, DETect fits into the monitoring stage of a MAPE loop, so planning and executing comprehensive behavioural adaptations is beyond the scope of this research. Nevertheless, DETect will be evaluated in this context in future work. In particular, the affect such adaptations may have on the validity of the model selected by DETect and whether an additional initialisation phase is required post-adaptation, will be analysed.

Static parameters: DETect is a novel algorithm with a high number of parameters used throughout its work flow. While varying all these parameters may impact on performance, the intention of the evaluation described here was to determine the feasibility of DETect as a decentralised approach to emergence detection. As a result, a decision was made to keep a number of parameters static once an acceptable level was found. Of note is the decision on the CUSUM variables h and k. Changing these parameter values is likely to greatly impact the sensitivity of DETect's feedback detection algorithm. The values chosen achieved a reasonable performance and allowed a focus on analysis of the parameters that were unique to DETect. This limitation could be compounded if applying DETect where there is little knowledge of either the domain or the potential emergents that can occur. As a result, this area is highlighted as possible future work in Section 6.3, to improve the general applicability of DETect.

5.6 Summary

This chapter presented a simulation-based case study to evaluate how well DETect achieves its objective of emergence detection in CAS. The case study included three multi-agent models of systems that exhibit spatial emergence, flocking, pedestrian counter-flow and traffic. Results demonstrate that DETect is an effective algorithm to achieve decentralised emergence detection in such systems. DETect's model selection performed well across all three models, successfully reducing the number of variables that needed to be monitored and selecting a higher number of useful variables compared to non-useful. In terms of detection, strong performance was achieved in both the pedestrian and traffic models during both emergence formation and evaporation with both the feedback detection and consensus formation generating a statistically higher number of events compared to periods without no change. In the flocking model, this performance was only achieved during emergence formation with DETect unable to effectively detect the evaporation of the flocking behaviour in the system. Although, this indicates that further work is required to improve the generalisability of DETect, overall performance supports the hypothesis that the presence of emergence changes an agents statistical relationship with its environment, providing a means of facilitating decentralised detection.

Chapter 6

Conclusion

This chapter summarises the thesis and its achievements and assesses its contribution to the state of the art in the domain of emergence detection. The chapter concludes with a discussion of the remaining open research issues that lend themselves to future work.

6.1 Thesis Summary

This thesis presented DETect, a novel distributed algorithm for decentralised emergence detection in complex adaptive systems.

Introduction: Chapter 1 motivated this work by outlining that emergence in complex adaptive systems can be either harmful or beneficial to both, the system as a whole or the constituent agents. It was argued that the main challenges to detecting emergence in these systems arise from their decentralised and unpredictable nature coupled with the unpredictable and transient nature of emergent behaviour and properties. This chapter hypothesised that feedback from emergence to the constituent agents, through downward causation, presented an opportunity of enabling decentralised detection of emergence. The hypothesis was that: When emergence is forming or evaporating in a system, a significant proportion of the constituent agents will simultaneously experience a change in the statistical relationship between themselves and their environment. By sharing this experience, agents can collaboratively act as detectors of the emergent event.

Background and Related Work: Chapter 2 described the history of emergence as a concept and identified its characteristics. It discussed emergence in the context of complex adaptive systems and multi-agent systems and identified the types of emergence that are the focus of this thesis. The second half of the chapter reviewed the state of the art in emergence detection and prediction techniques. Existing approaches were categorised under three broad types i) variable based ii) formal language/model based and iii) event based. The analysis highlighted the gap for a decentralised detection technique that operated at run time, did not require detailed knowledge of the system or expected emergence at design time, and could address the dynamic and transient nature of emergence.

Design: Chapter 3 returned to the challenges outlined in Chapter 1 to derive a set of design objectives necessary for an effective emergence detection technique. These objectives stated that detection must be decentralised in the system and occur at run time. No agent involved in detection can have a global view, therefore requiring local information to be used followed by collaboration with other detector agents. In addition, the technique should not require detailed a priori description of the expect emergent behaviour and should be capable of detecting when emergence forms and when it evaporates. After this, the system model positioned the contribution to decentralised emergence detection made by this thesis and described the scope of and assumptions of this work. Next, a discussion of design alternatives led to a set of design decisions that constitute the main contribution of this thesis, the proposed distributed algorithm, DETect. DETect enables the constituent agents of the system to act as detectors of emergence by looking for the effects of downward causation from emergence using locally available information. It achieves this by modelling the statistical relationship between an agent, characterised by internal variables, and its environment, characterised by external variables. The set of variables that make up this model are autonomously chosen by DETect at run time using LASSO. The relationship is analysed and monitored over time, with Cumulative Sum (CUSUM) used to detect when a significant change has occurred. Once a change is detected, agents share their experience using a distributed consensus protocol to determine if other agents are simultaneously experiencing a similar change. If a sufficient proportion of agents agree that a change is occurring, DETect concludes that an emergent event is occurring and a detection event is generated. Collectively, these design decisions support decentralised detection of the formation and evaporation of emergence in complex adaptive systems.

Implementation and Simulation Environment: Chapter 4 presented the implementation of

DETect and simulation environment using NetLogo, Java and R. It described three multi-agent models, **flocking, pedestrian counter-flow and traffic**, outlining how each model exhibits emergence and how DETect was integrated into each model's agents. These models provided a prototype, serving as a basis for evaluating DETect's performance.

Evaluation: Chapter 5 evaluated how well DETect achieves the overall objective of detecting emergence in complex adaptive systems. It presented a case study composed of three stages, each designed to evaluate one of DETect's core functionalities; model selection, feedback detection and emergence detection through consensus formation among agents. This evaluation was carried out using simulations of the three models presented in Chapter 4. The results demonstrated that DETect effectively detected both the formation and evaporation of emergence in both the traffic and pedestrian models with a statistically higher number of detection events generated during these periods compared to non-change periods. However, for the flocking model, DETect was only effective during emergence formation with evaporation periods not being detected by the algorithm. For model selection, DETect performed well across all three models, successfully reducing the number of variables that need to be monitored and simultaneously removing non-useful variables in larger numbers compared to useful variables.

6.2 Discussion

Figure 6.1 summarises the contributions made by DETect in the context of the challenges for emergence detection in complex adaptive systems (*cf.,* Section 1.3). The primary contribution is that *DETect is a mechanism for decentralised detection of emergence* by the constituent agents of the system. This property reflects the decentralised nature of the systems that generate emergence and is in contrast to existing detection techniques that depend on centralised architectures to varying degrees (*cf.,* Section 2.3). DETect achieves this by exploiting the concept of downward causation, where feedback from emergence constrains the agents at the micro-level of the system. This thesis demonstrates that this feedback results in a different statistical relationship between the agent and its environment when emergence is present compared to when there is no emergence in the system. By detecting these statistical changes and sharing information with each other, agents can collaboratively detect when emergence forms and evaporates in the system, creating the possibility for action to be taken to either leverage or mitigate the effects of the

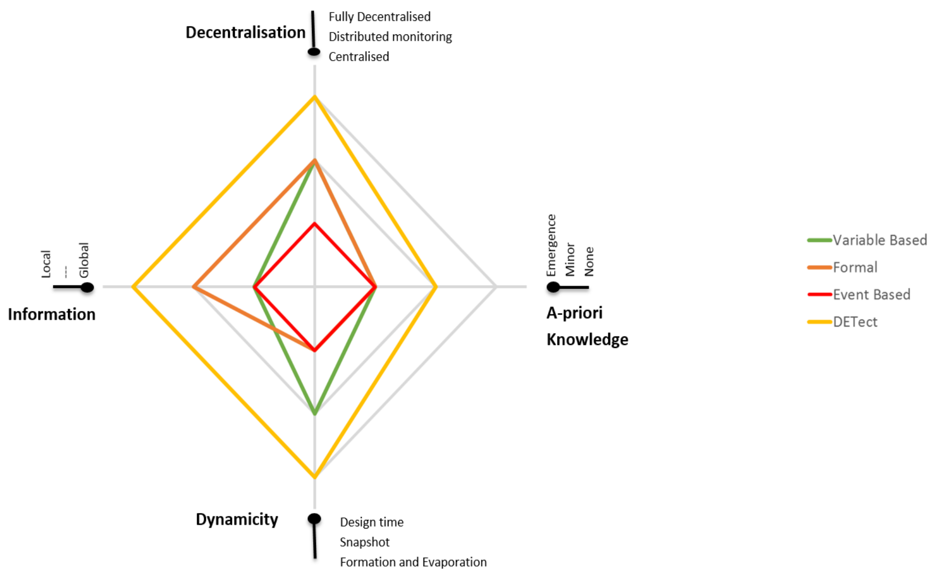

Fig. 6.1: ***Summary of DETect's contribution to knowledge*** *compared to the existing state of the art.*

emergence. To the best of our knowledge, this decentralised feedback-based approach is unique in the field of emergence detection.

A second contribution is to *reduce the need for design time knowledge of the specific type of emergence* that is expected in the system. This is in contrast to existing detection approaches with variable-based approaches requiring advance knowledge of the system-wide variables that describe emergence [Seth, 2008, Niazi and Hussain, 2011], and formal language and event-based approaches requiring the expected emergent behaviour and properties to be described [Ciancia et al., 2014, De Angelis and Di Marzo Serugendo, 2015]. DETect autonomously selects what variables to monitor at run time from the set of variables already available to the agent. These variables are used to model the relationship between the agent and its environment with this model used to facilitate detection of feedback from emergence. As a result, the requirement for specific design time knowledge of the emergent behaviour is removed. However, design time input is not completely eliminated as DETect uses a number of parameters throughout its work flow that affect performance and sensitivity. The evaluation carried out in this thesis identified which of these parameters have the greatest impact, however additional work is required to investigate a heuristics approach for setting these parameters in different types of systems (*cf.,* Section 6.3).

Finally, *DETect addresses the transient nature of emergence*, which forms and evaporates as the system evolves over time. This property of emergence and the timeliness of detection are not considered by existing approaches which evaluate a static snapshot of the systems evolution to determine if it contains emergence. This is in part due to their centralised architecture and need for system-wide properties to be calculated which takes time to accomplish. DETect uses a number of sliding observation windows throughout its work-flow to examine consecutive temporal windows of the systems evolution. Moreover, analysis is performed by each agent in the system, removing the need for system-wide properties to be gathered. This allows changes to be detected when they occur and facilitates detection of both the formation and evaporation of emergence in the system. This improves the timeliness of detection and allows an appropriate reaction to be made to the emergence while the macro-level state of the system is still in flux.

6.3 Future Work

Notwithstanding its contribution to knowledge, this thesis serves as a starting point for further investigation in the following areas:

Generalisability: Although this thesis has demonstrated the feasibility of the DETect approach, further research can yield improvements in the general applicability of the algorithm. First, DETect is composed of three functional units each with a set of parameters e.g., window sizes, thresholds etc., that need to be set appropriately. The evaluation presented in this thesis identified an acceptable level for a number of these parameters in the context of the simulation models. However, further work is needed to provide stronger guidelines for suitable ranges or to investigate techniques to allow DETect to set these parameters automatically. This is particularly important in cases where DETect may be used when there is little detailed knowledge of either the target domain or potential emergent behaviours that could arise. Next, the evaluation demonstrated that DETect did not perform well during the evaporation phase of the flocking model. This suggests that DETect could be improved by evaluating it on a broader range of models, including models that exhibit emergence that is not spatial in nature.

Adaptation: DETect is located in the monitoring stage of the MAPE loop [Kephart and Chess, 2003], with emergence detection events sent to an interested party, such as an adaptation manager, who will react appropriately. A suitable response to these events is not considered by

DETect, and DETect does not provide information on whether the detected emergent event is positive or negative for the agent. In addition, this thesis did not consider the effect a substantial adaptation of the agent's behavioural rules could have on DETect, especially when such an adaptation adds or removes variables. Future work may examine DETect within this broader adaptation contexts with particular emphasis on changes to the set of internal and external variables that may occur following an adaptation by the agent or a change in the environment. Such changes may render DETect's relationship model obsolete, requiring a new model to be selected. This may require a hybrid stage in DETect's work flow, where the model is updated gradually, allowing DETect to remain operational and prevent a complete reinitialisation. In terms of supplying semantic information to the adaptation manager on whether the emergent event is considered good or bad for the agent, it is possible that this information could also be incorporated into future versions of DETect. One potential way of achieving this is by monitoring how successful an agent has been in achieving its goals in the recent history of the system evolution.

More sophisticated modelling: The decision to use a multiple linear regression as the core model of the agent's relationship with its environment means that some accuracy has been traded for model efficiency. It is possible this may result in some subtle changes in non-linear relationships between variables being missed by DETect or that an apparent change occurs when, in reality, the relationship hasn't changed. The CUSUM change detection and distributed consensus algorithms are designed to cope with such inaccuracies. However, a more sophisticated model of the relationship between variables may yield greater accuracy in both feedback detection and emergence detection. This would require more computational expense during DETect's initialisation and model selection phase to explore alternative models. In addition, the requirement that model selection happen autonomously at run time means that there may be limits to how accurate the model can ever ultimately be.

Real-time systems: DETect is designed to operate in Complex Adaptive Systems, detecting the formation and evaporation of emergence at run time. The simulation models used in the evaluation case study (*cf.,* Chapter 5) use discrete time with each agent's clock assumed to be synchronised as part of the system model (*cf.,* Section 3.2.3). Therefore, agents periodically execute tasks such as observing the environment, detecting change using CUSUM, finding a partner and gossiping etc., concurrently during a single discrete time unit. In practise this assumption may limit DETect's effectiveness if deployed to a real-time system where agents have different computational power and DETect's components takes longer than a single unit of discrete time (whatever this may be) to complete these tasks. Of particular significance is the fundamental importance of agents simultaneously detecting feedback from emergence that is the core of DETect's approach. The current implementation includes a parameter called *Feedback Detection Memory* which facilitates this by allowing agents to remember feedback detection for a certain period of time. However, determining whether this is sufficient for real-time systems or whether more sophisticated solutions such as parallelization are required motivates additional research in this area.

6.4 Final Remark

This thesis investigated how to facilitate decentralised detection of emergence at run time in Complex Adaptive Systems, without detailed design time knowledge of the expected emergent behaviour or properties. Emergence detection is a hard problem however, the proposed distributed algorithm, DETect, was designed to allow the agents of the system to collaboratively act as detectors by detecting feedback from emergence that manifests as changes in the statistical relationship between the agent and its environment. A case study of three spatial multi-agent models demonstrated the feasibility of the DETect approach. Additional research may yield further improvement in the general applicability and accuracy of DETect. It is hoped that the findings of this thesis offer a new perspective on emergence detection and encourage further research in this area.

Appendix A

Identifying Formation And Evaporation of Emergence

This appendix describes the heuristic process that is applied to the time series of the objective measure of emergence following each simulation run, in order to identify the periods when emergence formed and evaporated. These periods are used to determine when DETect should detect generate feedback and emergence detection events. The process described here is implemented in R and uses the *pracma* and *signal* packages.

The same process is applied to the output from each of the 3 models. An illustrative example from the Traffic model is used to describe each step of the process.

A.1 Starting Point

The raw time series of the objective measure, A.1 is extremely noisy, which makes it very difficult to objectively identify when emergence formed and evaporated throughout the run. In this model, emergence formation results in the objective measure increasing, while evaporation results in the objective measure decreasing.

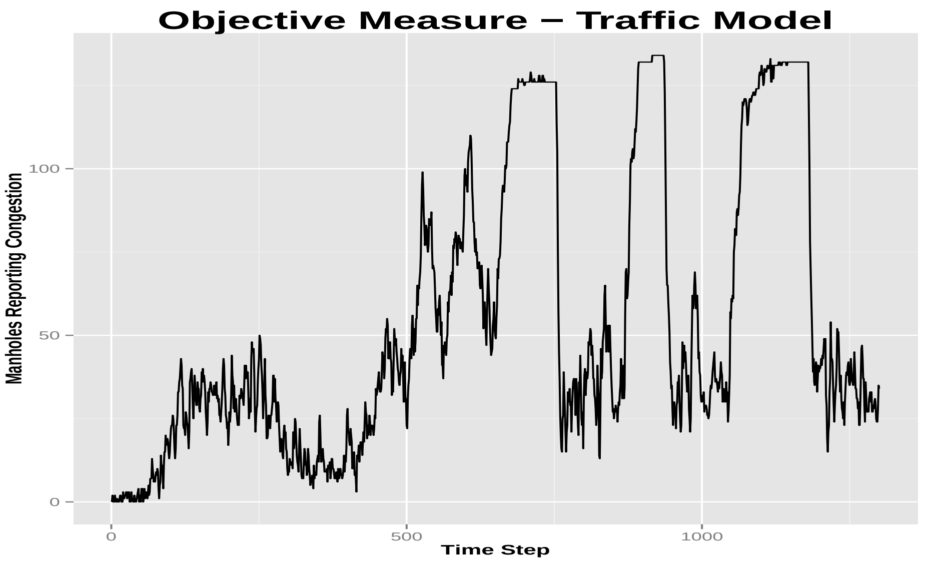

Fig. A.1: ***Start Point - raw time series:*** *An unprocessed time series of the objective measure of emergence from the Traffic model.*

A.2 Step 1 - Remove Noise

The first step in the heuristic process is to smooth the time-series using a low pass filter to remove the noise. The low pass filter removes small variations under a certain frequency in a signal and accentuates larger ones. The start of the time series is padded with 100 repetitions of its first value in order to prevent an edge effect occurring as the filter is a forward-backward filter. These are removed once the filter has been applied. A Butterworth filter (order = 2, critical frequency = 1/30) is applied to the raw time-series [Parks and Burrus, 1987]. The parameters of this filter were tuned following iterative experimentation with different values. Note, that the critical frequency that is chosen highly depends on the characteristics of the raw data. The result of the filter is illustrated in Figure A.2 and demonstrate that the jagged edges have been removed.

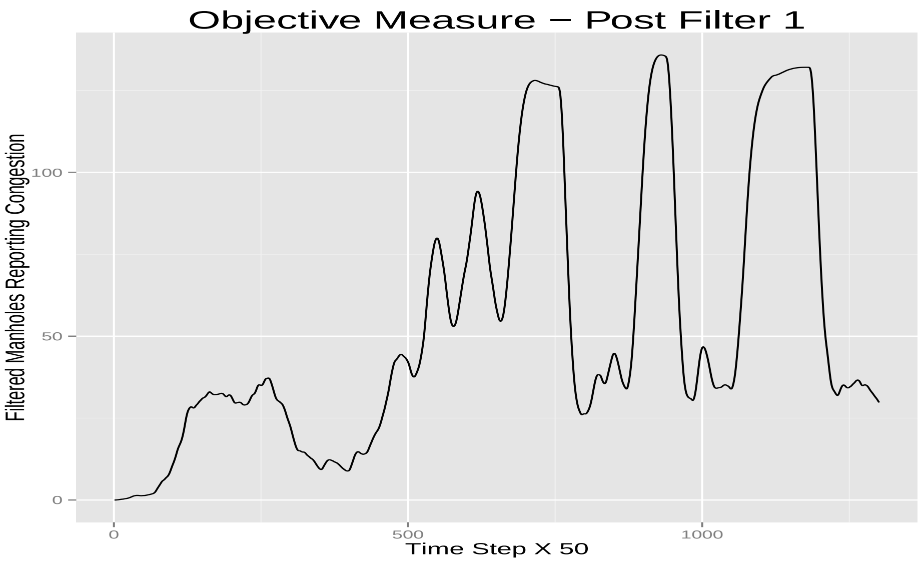

Fig. A.2: ***Step 1 - Remove Noise:*** *The time series of the objective measure following a low pass filter to remove noise.*

A.3 Step 2 - Finding Change

The next step is to find where change has occurred in the time series. By this we mean instances where the series either increased or decreased with respect to previous values. This is achieved by applying a 1st order difference the output from Step 1. The result is illustrated in Figure A.3. Note that the Y axis has now changed, from the absolute value of the objective measure to the change in its value compared to previous values.

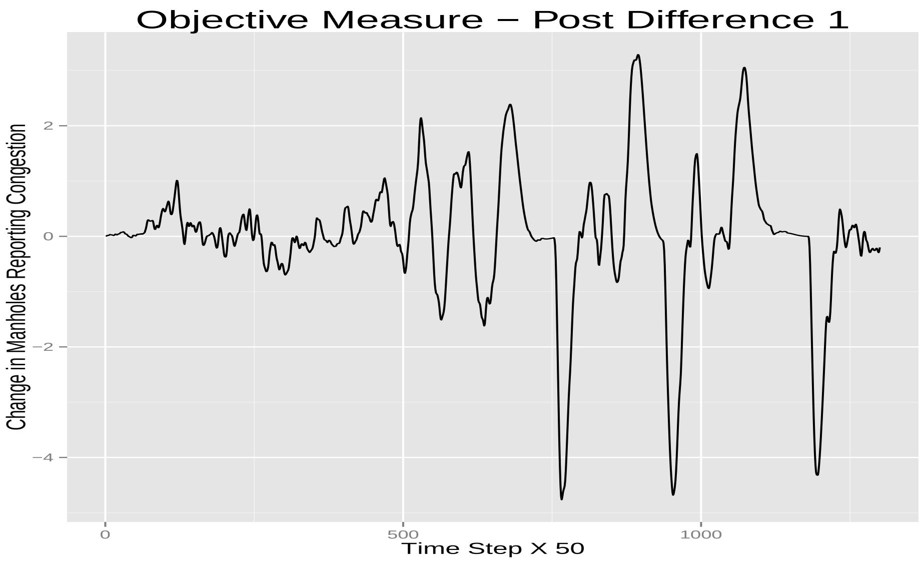

Fig. A.3: ***Step 2 - Finding change:*** *The time series of the objective measure once the 1st order difference has been applied.*

A.4 Step 3 - Remove the noise from the difference

The output from the difference is very noisy, similar to the initial raw data. We are only interested in significant changes that occur during the time series so we apply another low pass filter at this point to remove small changes from the series. Once again we pad the data and apply a Butterworth filter (order = 2, critical frequency = 1/30) as in Step 1 above. The output from this filter is illustrated in Figure A.4.

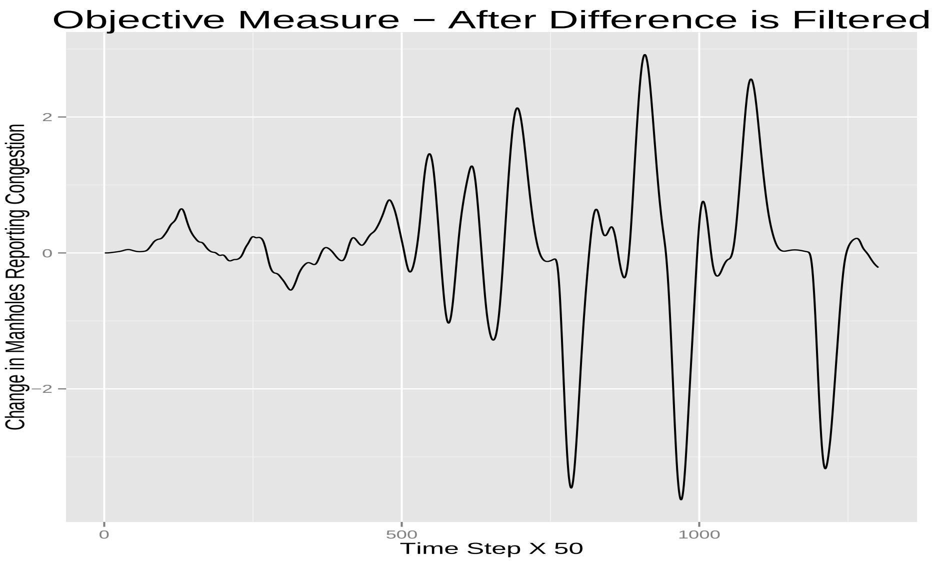

Fig. A.4: ***Step 3 - Remove the noise from difference:*** *The time series of the objective measure once the 1st order difference has been smoothed.*

A.5 Step 4 - Isolate positive and negative changes

The next step is to determine when both positive and negative changes occurred during the time series. At this point the time series contains a series of positive and negative numbers indicating if the objective measure increased or decreased at that time compared to earlier values in the time series.

First, we first split the original time series into two series, one containing only positive changes and the second containing only negative changes. Using the positive change series as an example, this is done by taking the original series and setting any negative values to 0. A similar process is applied to the negative series after which the negative series is inverted by multiplying it by -1. This has the effect of turning its "valleys" into peaks. The output of this step is illustrated by Figure A.5 and Figure A.6.

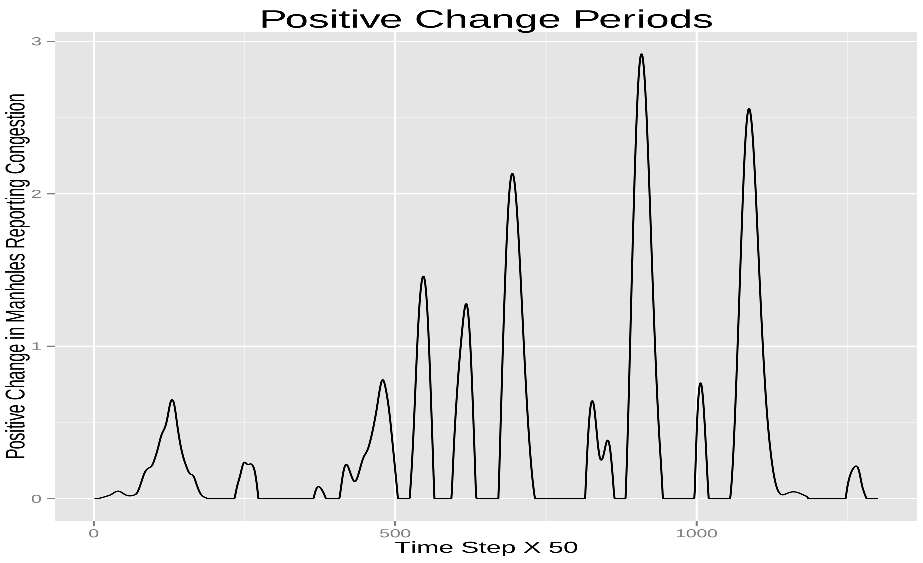

Fig. A.5: ***Step 4 - Positive changes:*** *s The time series of the positive change time series.*

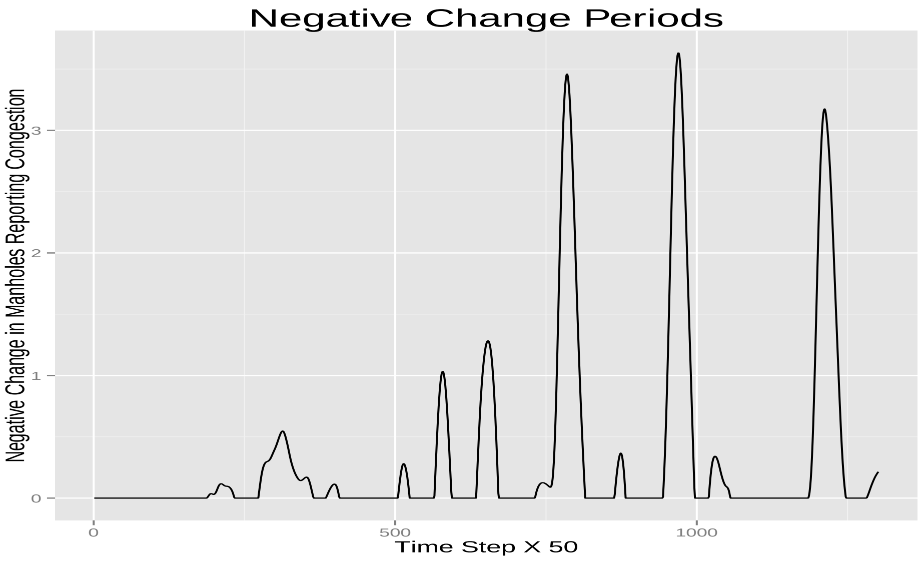

Fig. A.6: ***Step 4 - Negaitve changes:*** *The time series of the negative change time series.*

A.6 Step 5 - Determine when large changes begin and end

The final step is applied to each time-series to identify when large changes occurred. We use the *findpeaks* function of the *pracma* library which returns the location of the peaks in a time series of data. We specify a minimum peak height of twice the average absolute change across both the positive and negative change series. Once we have identified the time steps at which the peaks occurred, we find when the change associated with that peak started and ended by finding when the change was below 10% of the mean absolute change. This start and end time step correspond to the transition periods in the emergent state of the system. Periods identified in the positive time series are formation periods, while periods from the negative time series are evaporation periods. The results of the entire process is shown in Figure A.7, with the original time series overlaid on top of the identified formation periods (green) and evaporation periods (yellow).

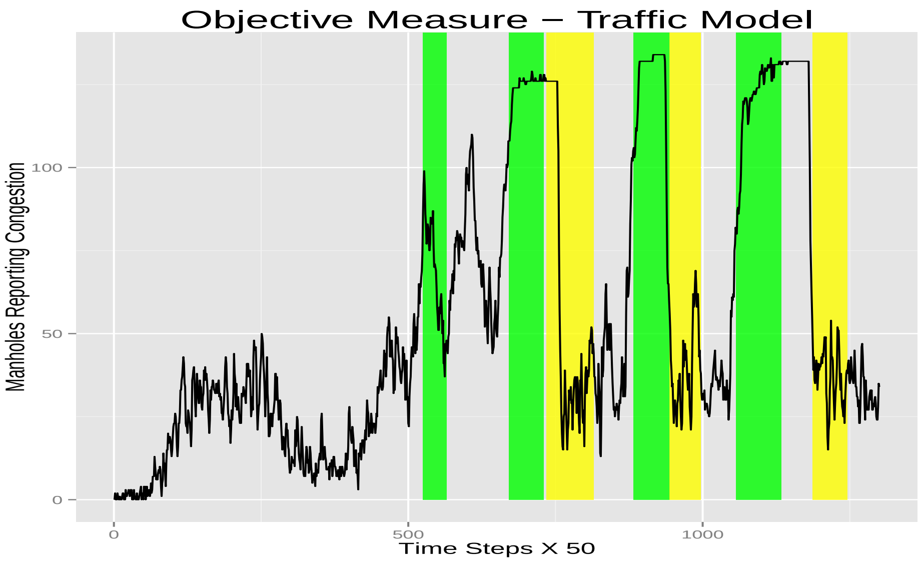

Fig. A.7: ***Step 5 - Identified transition periods:*** *Original time series with identified formation and evaporation periods.*

Appendix B

Stage 2: Change Detection ANOVA and Student's T-Tests

This appendix presents the analysis of variance (ANOVA) and Student's T-Tests undertaken to evaluate the performance of DETect in Stage 2 (change detection) of the case study described in Chapter 5 of this thesis. For each of the three models used in the study, evaluation is undertaken based on the number of detection events generated by DETect during formation, evaporation and non-change periods.

The Student's T-Tests performed are one-tailed tests, in each case to determine if the number of events generated by DETect during formation and evaporation periods is statistically significantly higher compared to non-change periods.

B.1 Flocking Model

Analysis of Variance

Table B.1: Stage 2 - Flocking Model Formation Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
CUSUMWIN	2	685.98	342.99	162.36	0.0000
REGWIN	1	1262.88	1262.88	597.81	0.0000
CUSUMWIN:REGWIN	2	283.90	141.95	67.19	0.0000
Residuals	54	114.08	2.11		

Table B.2: Stage 2 - Flocking Model Evaporation Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
CUSUMWIN	2	2817.77	1408.89	221.05	0.0000
REGWIN	1	2316.95	2316.95	363.52	0.0000
CUSUMWIN:REGWIN	2	2408.22	1204.11	188.92	0.0000
Residuals	54	344.18	6.37		

Table B.3: Stage 2 - Flocking Model Non-Change Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
CUSUMWIN	2	1483.28	741.64	2017.99	0.0000
REGWIN	1	1387.01	1387.01	3774.04	0.0000
CUSUMWIN:REGWIN	2	892.22	446.11	1213.85	0.0000
Residuals	54	19.85	0.37		

Student's T-Test

Table B.4: **Stage 2 - Flocking Model Student's T-Test** P-Values obtained for the **Flocking model** from one-tailed test to determine if DETect generated a statistically significantly higher number of detection events during transition periods compared to non-change periods.

CUSUMWIN	**REGWIN**	**Formation vs. Non-change Periods**	**Evaporation vs. Non-Change Periods**
High	High	0.9991	0.0003247
High	Low	3.875×10^{-05}	1
Low	High	1.836×10^{-05}	1
Low	Low	1.201×10^{-05}	0.9999
Medium	High	1.14×10^{-06}	0.9926
Medium	Low	4.402×10^{-06}	0.9996

B.2 Pedestrian Model

Analysis of Variance

Table B.5: Stage 2 - Pedestrian Model Non-Change Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
CUSUMWIN	2	119.68	59.84	17.53	0.0000
REGWIN	1	3730.52	3730.52	1093.03	0.0000
CUSUMWIN:REGWIN	2	77.65	38.83	11.38	0.0001
Residuals	54	184.30	3.41		

Table B.6: Stage 2 - Pedestrian Model Evaporation Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
CUSUMWIN	2	219.62	109.81	24.45	0.0000
REGWIN	1	3724.39	3724.39	829.12	0.0000
CUSUMWIN:REGWIN	2	167.48	83.74	18.64	0.0000
Residuals	54	242.57	4.49		

Table B.7: Stage 2 - Pedestrian Model Non-Change Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
CUSUMWIN	2	0.62	0.31	7.61	0.0012
REGWIN	1	40.87	40.87	994.80	0.0000
CUSUMWIN:REGWIN	2	0.33	0.17	4.06	0.0227
Residuals	54	2.22	0.04		

Student's T-Test

Table B.8: **Stage 2 - Pedestrian Model Student's T-Test** P-Values obtained for the **Pedestrian model** from one-tailed test to determine if DETect generated a statistically significantly higher number of detection events during transition periods compared to non-change periods.

CUSUMWIN	**REGWIN**	**Formation vs. Non-change Periods**	**Evaporation vs. Non-Change Periods**
High	High	6.003×10^{-09}	4.587×10^{-08}
High	Low	3.329×10^{-08}	6.576×10^{-08}
Low	High	3.461×10^{-09}	8.511×10^{-09}
Low	Low	2.423×10^{-06}	3.044×10^{-05}
Medium	High	1.259×10^{-08}	4.49×10^{-09}
Medium	Low	1.802×10^{-07}	1.085×10^{-05}

B.3 Traffic Model

Analysis of Variance

Table B.9: Stage 2 - Traffic Model Formation Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
CUSUMWIN	2	7360.04	3680.02	2.60	0.0835
REGWIN	1	490912.80	490912.80	346.87	0.0000
CUSUMWIN:REGWIN	2	6480.80	3240.40	2.29	0.1111
Residuals	54	76424.51	1415.27		

Table B.10: Stage 2 - Traffic Model Evaporation Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
CUSUMWIN	2	11369.38	5684.69	1.66	0.1999
REGWIN	1	370324.28	370324.28	108.08	0.0000
CUSUMWIN:REGWIN	2	9413.00	4706.50	1.37	0.2619
Residuals	54	185020.63	3426.31		

Student's T-Test

Table B.11: Stage 2 - Traffic Model Non-Change Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
CUSUMWIN	2	506.95	253.47	0.60	0.5501
REGWIN	1	180535.46	180535.46	430.49	0.0000
CUSUMWIN:REGWIN	2	348.33	174.16	0.42	0.6622
Residuals	54	22646.32	419.38		

Table B.12: **Stage 2 - Traffic model Student's T-Test** P-Values obtained for the **Traffic model** from one-tailed Student's T-Test to determine if DETect generated a statistically significantly higher number of detection events during transition periods compared to non-change periods.

CUSUMWIN	**REGWIN**	**Formation vs. Non-change Periods**	**Evaporation vs. Non-Change Periods**
High	High	0.0002982	0.01641
High	Low	0.01376	0.04474
Low	High	0.002883	0.03196
Low	Low	0.03756	0.1521
Medium	High	0.0020701	0.2291
Medium	Low	0.003588	0.01873

Appendix C

Stage 3: Consensus Formation ANOVA and Student's T-Tests

This appendix presents an ANOVA and Student's T-Tests undertaken to evaluate DETect's consensus formation functionality for each of the 3 models included in the case study described in Chapter 5 of this thesis. For each model, a different ANOVA is presented for formation, evaporation and non-change periods with the average number of emergence detection events generated every 50 time-steps constituting the response variable.

The Student's T-Tests performed are one-tailed tests, in each case to determine if the number of events generated by DETect during formation and evaporation periods is statistically significantly higher compared to non-change periods.

C.1 Flocking Model

ANOVA

Table C.1: Stage 3 - Flocking Model Formation Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
NeighMin	1	19.82	19.82	0.17	0.6792
NeighMax	1	1.28	1.28	0.01	0.9163
Threshold	1	21177.18	21177.18	184.41	0.0000
Memory	1	59015.66	59015.66	513.91	0.0000
NeighMin:NeighMax	1	579.80	579.80	5.05	0.0281
NeighMin:Threshold	1	50.89	50.89	0.44	0.5080
NeighMax:Threshold	1	147.98	147.98	1.29	0.2605
NeighMin:Memory	1	108.56	108.56	0.95	0.3346
NeighMax:Memory	1	0.01	0.01	0.00	0.9913
Threshold:Memory	1	373.50	373.50	3.25	0.0760
NeighMin:NeighMax:Threshold	1	32.13	32.13	0.28	0.5986
NeighMin:NeighMax:Memory	1	8.71	8.71	0.08	0.7839
NeighMin:Threshold:Memory	1	300.41	300.41	2.62	0.1107
NeighMax:Threshold:Memory	1	26.93	26.93	0.23	0.6299
NeighMin:NeighMax:Threshold:Memory	1	110.76	110.76	0.96	0.3297
Residuals	64	7349.54	114.84		

Table C.2: Stage 3 - Flocking Model Evaporation Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
NeighMin	1	6280.28	6280.28	32.73	0.0000
NeighMax	1	219.88	219.88	1.15	0.2885
Threshold	1	21486.82	21486.82	111.97	0.0000
Memory	1	24574.88	24574.88	128.06	0.0000
NeighMin:NeighMax	1	521.78	521.78	2.72	0.1041
NeighMin:Threshold	1	2335.24	2335.24	12.17	0.0009
NeighMax:Threshold	1	92.09	92.09	0.48	0.4910
NeighMin:Memory	1	4620.50	4620.50	24.08	0.0000
NeighMax:Memory	1	99.70	99.70	0.52	0.4737
Threshold:Memory	1	10826.08	10826.08	56.41	0.0000
NeighMin:NeighMax:Threshold	1	23.46	23.46	0.12	0.7278
NeighMin:NeighMax:Memory	1	478.42	478.42	2.49	0.1193
NeighMin:Threshold:Memory	1	1384.07	1384.07	7.21	0.0092
NeighMax:Threshold:Memory	1	164.13	164.13	0.86	0.3585
NeighMin:NeighMax:Threshold:Memory	1	15.24	15.24	0.08	0.7790
Residuals	64	12281.93	191.91		

Table C.3: Stage 3 - Flocking Model Non-Change Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
NeighMin	1	36.07	36.07	1.21	0.2757
NeighMax	1	26.64	26.64	0.89	0.3483
Threshold	1	14036.76	14036.76	470.30	0.0000
Memory	1	15469.62	15469.62	518.31	0.0000
NeighMin:NeighMax	1	21.85	21.85	0.73	0.3954
NeighMin:Threshold	1	44.19	44.19	1.48	0.2281
NeighMax:Threshold	1	16.52	16.52	0.55	0.4596
NeighMin:Memory	1	101.05	101.05	3.39	0.0704
NeighMax:Memory	1	4.07	4.07	0.14	0.7130
Threshold:Memory	1	5037.40	5037.40	168.78	0.0000
NeighMin:NeighMax:Threshold	1	34.09	34.09	1.14	0.2892
NeighMin:NeighMax:Memory	1	53.18	53.18	1.78	0.1866
NeighMin:Threshold:Memory	1	62.84	62.84	2.11	0.1517
NeighMax:Threshold:Memory	1	51.43	51.43	1.72	0.1940
NeighMin:NeighMax:Threshold:Memory	1	33.80	33.80	1.13	0.2912
Residuals	64	1910.17	29.85		

Student's T-Test

Table C.4: **Stage 3 - Flocking Model Neighbourhood Factors Student's T-Test** P-Values obtained for the **Flocking model** from one-tailed test to determine if DETect generated a statistically significantly higher number of detection events during transition periods compared to non-change periods.

NeighMIN	**NeighMax**	**Formation vs. Non-change Periods**	**Evaporation vs. Non-Change Periods**
High	High	1.675×10^{-07}	0.007276
High	Low	5.802×10^{-07}	0.05446
Low	High	9.057×10^{-08}	0.999
Low	Low	9.962×10^{-10}	0.9991

Table C.5: **Stage 3 - Flocking Model Threshold & Memory Factors Student's T-Test** P-Values obtained for the **Flocking model** from one-tailed test to determine if DETect generated a statistically significantly higher number of detection events during transition periods compared to non-change periods.

Threshold	**Memory**	**Formation vs. Non-change Periods**	**Evaporation vs. Non-Change Periods**
High	High	1.249×10^{-12}	0.7699
High	Low	1.522×10^{-08}	1
Low	High	4.03×10^{-11}	0.0409
Low	Low	3.261×10^{-13}	0.9026

C.2 Pedestrian Model

ANOVA

Table C.6: Stage 3 - Pedestrian Model Formation Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
NeighMin	1	1153.71	1153.71	2.66	0.1076
NeighMax	1	45.74	45.74	0.11	0.7463
Threshold	1	123473.67	123473.67	285.07	0.0000
Memory	1	50178.68	50178.68	115.85	0.0000
NeighMin:NeighMax	1	160.64	160.64	0.37	0.5447
NeighMin:Threshold	1	2137.25	2137.25	4.93	0.0299
NeighMax:Threshold	1	0.88	0.88	0.00	0.9642
NeighMin:Memory	1	219.62	219.62	0.51	0.4790
NeighMax:Memory	1	999.16	999.16	2.31	0.1337
Threshold:Memory	1	31532.60	31532.60	72.80	0.0000
NeighMin:NeighMax:Threshold	1	37.60	37.60	0.09	0.7692
NeighMin:NeighMax:Memory	1	53.11	53.11	0.12	0.7274
NeighMin:Threshold:Memory	1	615.04	615.04	1.42	0.2378
NeighMax:Threshold:Memory	1	482.14	482.14	1.11	0.2954
NeighMin:NeighMax:Threshold:Memory	1	300.14	300.14	0.69	0.4083
Residuals	64	27720.15	433.13		

Table C.7: Stage 3 - Pedestrian Model Evaporation Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
NeighMin	1	1813.07	1813.07	3.38	0.0705
NeighMax	1	29.04	29.04	0.05	0.8167
Threshold	1	113152.50	113152.50	211.11	0.0000
Memory	1	57749.13	57749.13	107.74	0.0000
NeighMin:NeighMax	1	77.18	77.18	0.14	0.7056
NeighMin:Threshold	1	1150.03	1150.03	2.15	0.1479
NeighMax:Threshold	1	222.64	222.64	0.42	0.5216
NeighMin:Memory	1	136.94	136.94	0.26	0.6150
NeighMax:Memory	1	426.91	426.91	0.80	0.3755
Threshold:Memory	1	5254.72	5254.72	9.80	0.0026
NeighMin:NeighMax:Threshold	1	239.35	239.35	0.45	0.5064
NeighMin:NeighMax:Memory	1	151.77	151.77	0.28	0.5965
NeighMin:Threshold:Memory	1	122.19	122.19	0.23	0.6347
NeighMax:Threshold:Memory	1	2.88	2.88	0.01	0.9418
NeighMin:NeighMax:Threshold:Memory	1	290.39	290.39	0.54	0.4644
Residuals	64	34303.45	535.99		

Table C.8: Stage 3 - Pedestrian Model Non-Change Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
NeighMin	1	0.00	0.00	0.05	0.8324
NeighMax	1	0.05	0.05	2.79	0.0996
Threshold	1	0.84	0.84	47.27	0.0000
Memory	1	0.66	0.66	36.99	0.0000
NeighMin:NeighMax	1	0.01	0.01	0.36	0.5510
NeighMin:Threshold	1	0.01	0.01	0.42	0.5215
NeighMax:Threshold	1	0.06	0.06	3.51	0.0654
NeighMin:Memory	1	0.00	0.00	0.09	0.7689
NeighMax:Memory	1	0.05	0.05	2.64	0.1092
Threshold:Memory	1	0.30	0.30	17.00	0.0001
NeighMin:NeighMax:Threshold	1	0.02	0.02	1.15	0.2869
NeighMin:NeighMax:Memory	1	0.00	0.00	0.24	0.6252
NeighMin:Threshold:Memory	1	0.01	0.01	0.64	0.4282
NeighMax:Threshold:Memory	1	0.06	0.06	3.23	0.0771
NeighMin:NeighMax:Threshold:Memory	1	0.02	0.02	0.84	0.3622
Residuals	64	1.14	0.02		

Student's T-Test

Table C.9: **Stage 3 - Pedestrian Model Neighbourhood Factors Student's T-Test** P-Values obtained for the **Pedestrian model** from one-tailed test to determine if DETect generated a statistically significantly higher number of detection events during transition periods compared to non-change periods.

NeighMIN	**NeighMax**	**Formation vs. Non-change Periods**	**Evaporation vs. Non-Change Periods**
High	High	0.0004524	2.997×10^{-05}
High	Low	0.002279	1.426×10^{-05}
Low	High	0.0006545	0.0001217
Low	Low	0.0006328	6.53×10^{-05}

Table C.10: **Stage 3 - Pedestrian Model Threshold & Memory Factors Student's T-Test** P-Values obtained for the **Pedestrian model** from one-tailed test to determine if DETect generated a statistically significantly higher number of detection events during transition periods compared to non-change periods.

Threshold	**Memory**	**Formation vs. Non-change Periods**	**Evaporation vs. Non-Change Periods**
High	High	$3.969\times10^{\{}$-06}	$6.579\times10^{\{}$-11}
High	Low	0.007402	0.0001034
Low	High	$3.012\times10^{\{}$-12}	$6.818\times10^{\{}$-13}
Low	Low	$1.213\times10^{\{}$-10}	$1.506\times10^{\{}$-10}

C.3 Traffic Model

ANOVA

Table C.11: Stage 3 - Traffic Model Formation Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
NeighMin	1	48524.79	48524.79	1.23	0.2720
NeighMax	1	191966.92	191966.92	4.86	0.0311
Threshold	1	657754.31	657754.31	16.64	0.0001
Memory	1	712424.40	712424.40	18.03	0.0001
NeighMin:NeighMax	1	150926.53	150926.53	3.82	0.0551
NeighMin:Threshold	1	126225.53	126225.53	3.19	0.0787
NeighMax:Threshold	1	3510.27	3510.27	0.09	0.7667
NeighMin:Memory	1	6170.36	6170.36	0.16	0.6941
NeighMax:Memory	1	5594.66	5594.66	0.14	0.7080
Threshold:Memory	1	3543.37	3543.37	0.09	0.7656
NeighMin:NeighMax:Threshold	1	38138.85	38138.85	0.96	0.3296
NeighMin:NeighMax:Memory	1	16783.51	16783.51	0.42	0.5170
NeighMin:Threshold:Memory	1	14265.32	14265.32	0.36	0.5501
NeighMax:Threshold:Memory	1	6734.10	6734.10	0.17	0.6812
NeighMin:NeighMax:Threshold:Memory	1	115147.97	115147.97	2.91	0.0927
Residuals	64	2529528.29	39523.88		

Table C.12: Stage 3 - Traffic Model Evaporation Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
NeighMin	1	2028.34	2028.34	0.04	0.8446
NeighMax	1	76330.66	76330.66	1.46	0.2317
Threshold	1	271820.18	271820.18	5.19	0.0260
Memory	1	504684.40	504684.40	9.64	0.0028
NeighMin:NeighMax	1	34148.73	34148.73	0.65	0.4223
NeighMin:Threshold	1	145187.87	145187.87	2.77	0.1008
NeighMax:Threshold	1	1114.28	1114.28	0.02	0.8845
NeighMin:Memory	1	12283.60	12283.60	0.23	0.6298
NeighMax:Memory	1	346.02	346.02	0.01	0.9355
Threshold:Memory	1	33780.47	33780.47	0.65	0.4248
NeighMin:NeighMax:Threshold	1	54564.10	54564.10	1.04	0.3112
NeighMin:NeighMax:Memory	1	231727.67	231727.67	4.43	0.0393
NeighMin:Threshold:Memory	1	134474.46	134474.46	2.57	0.1140
NeighMax:Threshold:Memory	1	5251.68	5251.68	0.10	0.7525
NeighMin:NeighMax:Threshold:Memory	1	65571.71	65571.71	1.25	0.2673
Residuals	64	3351154.01	52361.78		

Student's T-Test

Table C.13: Stage 3 - Traffic Model Non-Change Periods ANOVA

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
NeighMin	1	49412.10	49412.10	6.78	0.0115
NeighMax	1	776.47	776.47	0.11	0.7452
Threshold	1	351344.45	351344.45	48.18	0.0000
Memory	1	304179.19	304179.19	41.71	0.0000
NeighMin:NeighMax	1	1222.91	1222.91	0.17	0.6835
NeighMin:Threshold	1	13463.51	13463.51	1.85	0.1790
NeighMax:Threshold	1	239.43	239.43	0.03	0.8568
NeighMin:Memory	1	2112.27	2112.27	0.29	0.5923
NeighMax:Memory	1	12732.16	12732.16	1.75	0.1911
Threshold:Memory	1	37198.32	37198.32	5.10	0.0273
NeighMin:NeighMax:Threshold	1	313.35	313.35	0.04	0.8364
NeighMin:NeighMax:Memory	1	510.90	510.90	0.07	0.7921
NeighMin:Threshold:Memory	1	1741.81	1741.81	0.24	0.6267
NeighMax:Threshold:Memory	1	1400.19	1400.19	0.19	0.6627
NeighMin:NeighMax:Threshold:Memory	1	22723.86	22723.86	3.12	0.0823
Residuals	64	466682.29	7291.91		

Table C.14: **Stage 3 - Traffic Model Neighbourhood Factors Student's T-Test** P-Values obtained for the **Traffic model** from one-tailed test to determine if DETect generated a statistically significantly higher number of detection events during transition periods compared to non-change periods.

NeighMIN	**NeighMax**	**Formation vs. Non-change Periods**	**Evaporation vs. Non-Change Periods**
High	High	1.521×10^{-05}	6.82×10^{-05}
High	Low	0.0001913	0.0001256
Low	High	2.557×10^{-05}	0.03744
Low	Low	0.0006516	.0005213

Table C.15: **Stage 3 - Traffic Model Threshold & Memory Factors Student's T-Test** P-Values obtained for the **Traffic model** from one-tailed test to determine if DETect generated a statistically significantly higher number of detection events during transition periods compared to non-change periods.

Threshold	**Memory**	**Formation vs. Non-change Periods**	**Evaporation vs. Non-Change Periods**
High	High	0.0001319	$6.265\times10^{\{}$-05}
High	Low	0.0006799	0.001258
Low	High	0.0001522	0.006962
Low	Low	$1.637\times10^{\{}$-06}	0.0009383

Bibliography

[Abbott, 2006] Abbott, R. (2006). "Emergence explained: Abstractions: Getting epiphenomena to do real work". *Complexity*, 12(1), pp. 13–26. [Cited on page 13.]

[Adams and MacKay, 2007] Adams, R. P. and MacKay, D. J. (2007). "Bayesian online changepoint detection". *arXiv preprint arXiv:0710.3742*. [Cited on page 60.]

[Akaike, 1973] Akaike, H. (1973). "{Information theory and an extension of the maximum likelihood principle}". In Petrov, B. N. and Csaki, F., editors, *Second international symposium on information theory*, pages 267 – 281. Budapest: Academiai Kiado. [Cited on page 57.]

[Albert and Chib, 1997] Albert, J. and Chib, S. (1997). "Bayesian Tests and Model Diagnostics in Conditionally Independent Hierarchical Models". *Journal of the American Statistical Association*, 92(439), pp. 916–925. [Cited on page 57.]

[Alvarez-Molina et al., 2014] Alvarez-Molina, E. R.; Martinez, L. G.; Castanon-Puga, M.; and Rodriguez-Diaz, A. (2014). "A Neuro-Fuzzy System as a complex system of emergent behavior in organizations". In *2014 Second World Conference on Complex Systems (WCCS)*, pages 463–468. IEEE. [Cited on page 27.]

[Ambrose and Grasela, 2000] Ambrose, P. G. and Grasela, D. M. (2000). "The use of Monte Carlo simulation to examine pharmacodynamic variance of drugs: fluoroquinolone pharmacodynamics against Streptococcus pneumoniae". *Diagnostic microbiology and infectious disease*, 38(3), pp. 151–157. [Cited on page 29.]

[Anderson, 1999] Anderson, P. W. (1999). "The Eightfold Way to the theory of complexity: a prologue". In *Complexity*, pages 7–16. Perseus Books. [Cited on page 17.]

[Arel et al., 2010] Arel, I.; Liu, C.; Urbanik, T.; and Kohls, A. (2010). “Reinforcement learning-based multi-agent system for network traffic signal control”. *Intelligent Transport Systems, IET*, 4(2), pp. 128–135. [Cited on page 30.]

[Arthur et al., 1997] Arthur, W. B.; Durlauf, S.; and Lane, D. A. (1997). “Process and Emergence in the Economy”. *The Economy.* [Cited on page 24.]

[Bar-Yam, 2004] Bar-Yam, Y. (2004). “A mathematical theory of strong emergence using multiscale variety”. *Complexity*, 9(6), pp. 15–24. [Cited on pages 6 and 20.]

[Baxt, 1995] Baxt, W. G. (1995). “Application of artificial neural networks to clinical medicine”. *The lancet*, 346(8983), pp. 1135–1138. [Cited on page 27.]

[BBBike.org, 2014] BBBike.org (2014). “OSM extracts for New York”. [Cited on page 98.]

[Bedau, 2002] Bedau, M. (2002). “Downward causation and the autonomy of weak emergence”. *Principia: an international journal of epistemology*, 6(1), pp. 5–50. [Cited on pages 2, 7, 17, 18, 20, and 54.]

[Bedau, 1997] Bedau, M. A. (1997). “Weak emergence”. *Noûs*, 31(s11), pp. 375–399. [Cited on page 17.]

[Bernon et al., 2006] Bernon, C.; Chevrier, V.; Hilaire, V.; and Marrow, P. (2006). “Applications of self-organising multi-agent systems: An initial framework for comparison”. *Informatica*, 30(1). [Cited on page 30.]

[Bernon et al., 2005] Bernon, C.; Cossentino, M.; Gleizes, M.-P.; Turci, P.; and Zambonelli, F. (2005). “A study of some multi-agent meta-models”. In *Agent-Oriented Software Engineering V*, pages 62–77. Springer. [Cited on page 31.]

[Birdsey and Szabo, 2014] Birdsey, L. and Szabo, C. (2014). “An architecture for identifying emergent behavior in multi-agent systems”. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1455–1456. International Foundation for Autonomous Agents and Multiagent Systems. [Cited on page 37.]

[Bonabeau, 2002] Bonabeau, E. (2002). “Agent-based modeling: Methods and techniques for simulating human systems”. *Proceedings of the National Academy of Sciences*, 99(suppl 3), pp. 7280–7287. [Cited on page 29.]

[Booker et al., 1989] Booker, L. B.; Goldberg, D. E.; and Holland, J. H. (1989). "Classifier systems and genetic algorithms". *Artificial intelligence*, 40(1), pp. 235–282. [Cited on page 27.]

[Bouarfa et al., 2013] Bouarfa, S.; Blom, H. A. P.; Curran, R.; and Everdij, M. H. C. (2013). "Agent-based modeling and simulation of emergent behavior in air transportation". *Complex Adaptive Systems Modeling*, 1(1), pp. 1–26. [Cited on page 29.]

[Brownlee, 2007] Brownlee, J. (2007). "Complex adaptive systems". *Complex Intelligent Systems Laboratory, Centre for Information Technology Research, Faculty of Information Communication Technology, Swinburne University of Technology: Melbourne, Australia.* [Cited on page 25.]

[Bull, 2004] Bull, L. (2004). "Learning classifier systems: A brief introduction". In *Applications of Learning Classifier Systems*, pages 1–12. Springer. [Cited on pages 27 and 28.]

[Bull et al., 2004] Bull, L.; ShaAban, J.; Tomlinson, A.; Addison, J. D.; and Heydecker, B. G. (2004). "Towards distributed adaptive control for road traffic junction signals using learning classifier systems". In *Applications of Learning Classifier Systems*, pages 276–299. Springer. [Cited on page 28.]

[Cabri et al., 2011] Cabri, G.; Puviani, M.; and Zambonelli, F. (2011). "Towards a taxonomy of adaptive agent-based collaboration patterns for autonomic service ensembles". In *Collaboration Technologies and Systems (CTS), 2011 International Conference on*, pages 508–515. IEEE. [Cited on page 53.]

[Cagan and Ready, 1989] Cagan, R. L. and Ready, D. F. (1989). "The emergence of order in the Drosophila pupal retina". *Developmental biology*, 136(2), pp. 346–362. [Cited on page 12.]

[Capera et al., 2003] Capera, D.; George, J.-P.; Gleizes, M.-P.; and Glize, P. (2003). "The AMAS theory for complex problem solving based on self-organizing cooperative agents". In *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, pages 383–388. IEEE Comput. Soc. [Cited on page 31.]

[Carli et al., 2010] Carli, R.; Fagnani, F.; Frasca, P.; and Zampieri, S. (2010). "Gossip consensus algorithms via quantized communication". *Automatica*, 46(1), pp. 70–80. [Cited on page 63.]

[Chalmers, 2006] Chalmers, D. J. (2006). "Strong and Weak Emergence". *The reemergence of emergence: The Emergentist hypothesis from science to religion*, pages 244–256. [Cited on pages 7 and 17.]

[Chan, 2011] Chan, W. K. V. (2011). "Interaction metric of emergent behaviors in agent-based simulation". *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 357–368. [Cited on pages 6, 36, 38, 39, and 83.]

[Chatty et al., 2013] Chatty, A.; Gaussier, P.; Kallel, I.; Laroque, P.; Pirard, F.; and Alimi, A. M. (2013). "Evaluation of Emergent Structures in a "Cognitive" Multi-Agent System Based on On-line Building and Learning of a Cognitive Map". In *5th International Conference on Agents and Artificial Intelligence (ICAART)*, pages 269–275. [Cited on page 32.]

[Chen, Chih-Chun and Nagl, Sylvia B and Clack, 2008] Chen, Chih-Chun and Nagl, Sylvia B and Clack, C. D. (2008). "A method for validating and discovering associations between multi-level emergent behaviours in agent-based simulations". In *Agent and Multi-Agent Systems: Technologies and Applications*, pages 1–10. Springer. [Cited on pages 42 and 43.]

[Ciancia et al., 2014] Ciancia, V.; Latella, D.; Loreti, M.; and Massink, M. (2014). "Specifying and verifying properties of space". In *Theoretical Computer Science*, pages 222–235. Springer. [Cited on pages 6, 40, 42, and 146.]

[Cornell, 2015] Cornell (2015). "Conway's Game of Life". [Online; accessed 03-October-2015]. [Cited on page 26.]

[Das et al., 2014] Das, S.; Goswami, D.; Chatterjee, S.; and Mukherjee, S. (2014). "Stability and chaos analysis of a novel swarm dynamics with applications to multi-agent systems". *Engineering Applications of Artificial Intelligence*, 30, pp. 189–198. [Cited on page 30.]

[Datta et al., 2004] Datta, A.; Quarteroni, S.; and Aberer, K. (2004). "Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in wireless mobile ad-hoc networks". In *Semantics of a Networked World. Semantics for Grid Databases*, pages 126–143. Springer. [Cited on page 63.]

[De Angelis and Di Marzo Serugendo, 2015] De Angelis, F. L. and Di Marzo Serugendo, G. (2015). "A logic language for run time assessment of spatial properties in self-organizing systems". In *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 86–91. IEEE. [Cited on pages 6, 40, 41, and 146.]

[De Haan, 2006] De Haan, J. (2006). "How emergence arises". *Ecological Complexity*, 3(4), pp. 293–301. [Cited on pages 19, 20, 21, and 51.]

[De Wolf and Holvoet, 2005] De Wolf, T. and Holvoet, T. (2005). "Emergence Versus Self-organisation : Different Concepts but Promising When Combined". *Engineering self-organising systems*, pages 77–91. [Cited on pages 2, 5, 13, 16, 21, 22, 47, and 75.]

[De Wolf et al., 2005] De Wolf, T.; Samaey, G.; Holvoet, T.; and Roose, D. (2005). "Decentralised Autonomic Computing: Analysing Self-Organising Emergent Behaviour using Advanced Numerical Methods". In *Second International Conference on Autonomic Computing (ICAC'05)*, pages 52–63. IEEE. [Cited on pages 6 and 36.]

[Deguet et al., 2006] Deguet, J.; Demazeau, Y.; and Magnin, L. (2006). "Elements about the Emergence Issue: A Survey of Emergence Definitions". *Complexus*, 3(1-3), pp. 24–31. [Cited on page 2.]

[Dessalles et al., 2007] Dessalles, J. L.; Muller, J. P.; and Phan, D. (2007). "Emergence in multi-agent systems : conceptual and methodological issues". In Amblard, F & Phan, D., editor, *Agent-based modelling and simulation in the social and human sciences*, pages 327–355. Oxford, The Bardwell-Press. [Cited on page 30.]

[Dessalles and Phan, 2001] Dessalles, J.-L. and Phan, D. (2001). "Emergence in multi-agent systems : cognitive hierarchy , detection , and complexity reduction part I : methodological issues". In *Artificial Economics*, pages 147—-159. Springer. [Cited on pages 3 and 4.]

[Devaney et al., 1989] Devaney, R. L.; Devaney, L.; and Devaney, L. (1989). *An introduction to chaotic dynamical systems*, volume 13046. Addison-Wesley Reading. [Cited on page 16.]

[Di Marzo Serugendo et al., 2004] Di Marzo Serugendo, G.; Foukia, N.; Hassas, S.; Karageorgos, A.; Mostéfaoui, S. K.; Rana, O. F.; Ulieru, M.; Valckenaers, P.; and Van Aart, C. (2004). *Self-organisation: Paradigms and applications.* Springer. [Cited on page 22.]

[Di Marzo Serugendo et al., 2006] Di Marzo Serugendo, G.; Irit, M.-P.; and Karageorgos, A.

(2006). “Self-organisation and emergence in MAS: An overview”. *Informatica*, 30(1). [Cited on pages 2, 14, 22, 30, 31, and 47.]

[Dooley, 1996] Dooley, K. (1996). “Complex adaptive systems: A nominal definition”. *The Chaos Network*, 8(1), pp. 2–3. [Cited on page 23.]

[Dorigo et al., 2000] Dorigo, M.; Bonabeau, E.; and Theraulaz, G. (2000). “Ant algorithms and stigmergy”. *Future Generation Computer Systems*, 16(8), pp. 851–871. [Cited on pages 29 and 62.]

[Draper and Smith, 1981] Draper, N. R. and Smith, H. (1981). *Applied regression analysis 2nd ed.* New York New York John Wiley and Sons 1981. [Cited on page 55.]

[El-Hani and Pihlström, 2002] El-Hani, C. N. n. and Pihlström, S. (2002). “Emergence theories and pragmatic realism”. *Essays in Philosophy*, 3(2), pp. 3. [Cited on page 17.]

[Farmer and Foley, 2009] Farmer, J. D. and Foley, D. (2009). “The economy needs agent-based modelling”. *Nature*, 460(7256), pp. 685–686. [Cited on page 29.]

[Fernandez-Marquez et al., 2012] Fernandez-Marquez, J. L.; Di Marzo Serugendo, G.; Montagna, S.; Viroli, M.; and Arcos, J. L. (2012). “Description and composition of bio-inspired design patterns: a complete overview”. *Natural Computing*, 12(1), pp. 43–67. [Cited on pages 30 and 62.]

[Fisch et al., 2010] Fisch, D.; Janicke, M.; Sick, B.; and Muller-Schloer, C. (2010). “Quantitative Emergence – A Refined Approach Based on Divergence Measures”. In *2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, pages 94–103. IEEE. [Cited on pages 6 and 34.]

[Flake, 1998] Flake, G. W. (1998). *The computational beauty of nature: Computer explorations of fractals, chaos, complex systems, and adaptation.* MIT press. [Cited on page 2.]

[Fromm, 2005] Fromm, J. (2005). “Types and forms of emergence”. *arXiv preprint nlin/0506028.* [Cited on pages 2, 20, 21, 31, and 51.]

[Galván-López et al., 2014] Galván-López, E.; Taylor, A.; Clarke, S.; and Cahill, V. (2014). “Design of an automatic demand-side management system based on evolutionary algorithms”. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 525–530. ACM. [Cited on page 29.]

[Gardner, 1970] Gardner, M. (1970). "Mathematical games: The fantastic combinations of John Conways new solitaire game life". *Scientific American*, 223(4), pp. 120–123. [Cited on pages 20 and 26.]

[Gelernter and Carriero, 1992] Gelernter, D. and Carriero, N. (1992). "Coordination languages and their significance". *Communications of the ACM*, 35(2), pp. 96. [Cited on page 63.]

[Gell-Mann, 1994] Gell-Mann, M. (1994). "Complex Adaptive Systems". In Cowan, G.; Pines, D.; and Meltzer, D., editors, *Complexity: metaphors, models, and reality*, pages 17—-45. Westview Press. [Cited on pages 17, 23, 24, 25, and 50.]

[Gemmer et al., 2009] Gemmer, J.; Michel, M.; and Mahler, G. (2009). "Quantum Thermodynamcis-Emergence of Thermodynamic Behavior within Composite Quantum Systems". *Lecture Notes in Physics 2nd ed.(Springer, 2009).* [Cited on page 12.]

[Gershenson and Heylighen, 2003] Gershenson, C. and Heylighen, F. (2003). "When can we call a system self-organizing?". In *Advances in artificial life*, pages 606–614. Springer. [Cited on page 22.]

[Gleizes et al., 1999] Gleizes, M.-P.; Camps, V.; and Glize, P. (1999). "A theory of emergent computation based on cooperative self-organization for adaptive artificial systems". In *Fourth European Congress of Systems Science*, pages 20–24. [Cited on page 30.]

[Gleizes et al., 2008] Gleizes, M.-P.; Georg, J.-P.; and Capera, D. (2008). "Engineering Systems Which Generate Emergent Functionalities". *Engineering Environment-Mediated Multi-Agent Systems*, 5049/2008, pp. 58–75. [Cited on page 2.]

[Goh, 1995] Goh, A. (1995). "Back-propagation neural networks for modeling complex systems". *Artificial Intelligence in Engineering*, 9(3), pp. 143–151. [Cited on page 27.]

[Goldenberg et al., 2001] Goldenberg, J.; Libai, B.; and Muller, E. (2001). "Talk of the network: A complex systems look at the underlying process of word-of-mouth". *Marketing letters*, 12(3), pp. 211–223. [Cited on page 26.]

[Goldstein, 1999] Goldstein, J. (1999). "Emergence as a Construct: History and Issues". *Emergence*, 1(1), pp. 49–72. [Cited on pages 13, 14, 15, 16, 43, 47, and 48.]

[Goldstein, 2000] Goldstein, J. (2000). “Emergence: A Construct Amid a Thicket of Conceptual Snares”. *Emergence*, 2(1), pp. 5–22. [Cited on page 14.]

[Granger, 1969] Granger, C. W. (1969). “Investigating causal relations by econometric models and cross-spectral methods”. *Econometrica: Journal of the Econometric Society*, pages 424–438. [Cited on page 35.]

[Graphhopper, 2014] Graphhopper (2014). “GraphHopper”. [Cited on page 98.]

[Grassé, 1959] Grassé, P.-P. (1959). “La reconstruction du nid et les coordinations interindividuelles chezBellicositermes natalensis etCubitermes sp. la th{é}orie de la stigmergie: Essai d’interpr{é}tation du comportement des termites constructeurs”. *Insectes sociaux*, 6(1), pp. 41–80. [Cited on pages 29 and 62.]

[Grilo et al., 2002] Grilo, A.; Caetano, A.; and Rosa, A. (2002). “Immune system simulation through a complex adaptive system model”. In *Soft Computing and Industry*, pages 675–698. Springer. [Cited on page 26.]

[Grossman et al., 2009] Grossman, R.; Sabala, M.; Handley, M.; and Wilkinson, L. (2009). “Discovering Emergent Behavior From Network Packet Data : Lessons from the Angle Project”. *Next Generation of Data Mining*, pages 243—-260. [Cited on pages 6, 36, 37, and 60.]

[Haken, 1983] Haken, H. (1983). *Advanced synergetics*. Springer Berlin. [Cited on page 16.]

[Hawkins, 1987] Hawkins, D. M. (1987). “Self-starting CUSUM charts for location and scale”. *The Statistician*, pages 299–316. [Cited on pages 60 and 72.]

[Healy, 1987] Healy, J. D. (1987). “A note on multivariate CUSUM procedures”. *Technometrics*, 29(4), pp. 409–412. [Cited on page 60.]

[Helbing et al., 2002] Helbing, D.; Farkas, I. J.; Molnar, P.; and Vicsek, T. (2002). “Simulation of pedestrian crowds in normal and evacuation situations”. *Pedestrian and evacuation dynamics*, 21(2), pp. 21–58. [Cited on page 94.]

[Helbing and Molnar, 1998] Helbing, D. and Molnar, P. (1998). “Self-organization phenomena in pedestrian crowds”. *arXiv preprint cond-mat/9806152*. [Cited on page 94.]

[Heylighen, 2001] Heylighen, F. (2001). "The science of self-organization and adaptivity". *The encyclopedia of life support systems*, 5(3), pp. 253–280. [Cited on page 17.]

[History Of Entropy, 2015] History Of Entropy (2015). "History Of Entropy — Wikipedia, The Free Encyclopedia". [Online; accessed 29-September-2015]. [Cited on page 33.]

[Hoekstra et al., 2010] Hoekstra, A. G.; Kroc, J.; and Sloot, P. M. (2010). *Simulating complex systems by cellular automata.* Springer. [Cited on page 26.]

[Holland, 1976] Holland, J. H. (1976). "Adaptation". In *Progress in Theoretical Biology*, pages 263–293. Elsevier. [Cited on page 27.]

[Holland, 1980] Holland, J. H. (1980). "Adaptive algorithms for discovering and using general patterns in growing knowledge bases". *International Journal of Policy Analysis and Information Systems*, 4(3), pp. 245–268. [Cited on page 27.]

[Holland, 1992] Holland, J. H. (1992). "Complex adaptive systems". *Daedalus*, pages 17–30. [Cited on pages 1, 4, 17, 23, 47, and 50.]

[Holland, 1995] Holland, J. H. (1995). *Hidden order: How adaptation builds complexity.* Basic Books. [Cited on pages 23 and 28.]

[Holland, 2000] Holland, J. H. (2000). *Emergence: From chaos to order.* Oxford University Press. [Cited on pages 13, 14, and 30.]

[Holzer et al., 2008] Holzer, R.; De Meer, H.; and Bettstetter, C. (2008). *On autonomy and emergence in self-organizing systems.* Springer. [Cited on page 35.]

[Huttenlocher et al., 1993] Huttenlocher, D. P.; Klanderman, G.; Rucklidge, W. J.; et al. (1993). "Comparing images using the Hausdorff distance". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(9), pp. 850–863. [Cited on page 37.]

[Innes and Booher, 1999] Innes, J. E. and Booher, D. E. (1999). "Consensus building and complex adaptive systems: A framework for evaluating collaborative planning". *Journal of the American planning association*, 65(4), pp. 412–423. [Cited on page 2.]

[Jelasity et al., 2005] Jelasity, M.; Montresor, A.; and Babaoglu, O. (2005). "Gossip-based aggregation in large dynamic networks". *ACM Transactions on Computer Systems*, 23(3), pp. 219–252. [Cited on page 63.]

[Jennings, 2001] Jennings, N. R. (2001). “An agent-based approach for building complex software systems”. *Communications of the ACM*, 44(4), pp. 35–41. [Cited on page 29.]

[Johnson, 2006] Johnson, C. W. (2006). “What are emergent properties and how do they affect the engineering of complex systems?”. *Reliability Engineering & System Safety*, 91(12), pp. 1475–1481. [Cited on page 2.]

[Kauffman, 1995] Kauffman, S. (1995). *At home in the universe: The search for the laws of self-organization and complexity.* Oxford University Press, USA. [Cited on page 18.]

[Kauffman, 1993] Kauffman, S. A. (1993). *The origins of order: Self organization and selection in evolution.* Oxford university press. [Cited on page 28.]

[Kawahara et al., 2007] Kawahara, Y.; Yairi, T.; and Machida, K. (2007). “Change-point detection in time-series data based on subspace identification”. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 559–564. IEEE. [Cited on page 60.]

[Kempe et al., 2003] Kempe, D.; Dobra, A.; and Gehrke, J. (2003). “Gossip-based computation of aggregate information”. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 482–491. IEEE. [Cited on page 63.]

[Kephart and Chess, 2003] Kephart, J. and Chess, D. (2003). “The vision of autonomic computing”. *Computer*, 36(1), pp. 41–50. [Cited on pages 7, 53, 54, and 147.]

[Kim, 1992] Kim, J. (1992). “Downward causationin emergentism and nonreductive physicalism”. *Emergence or reduction*, pages 119–138. [Cited on page 17.]

[Kim, 1999] Kim, J. (1999). “Making sense of emergence”. *Philosophical studies*, 95(1), pp. 3–36. [Cited on page 17.]

[Koch and Laurent, 1999] Koch, C. and Laurent, G. (1999). “Complexity and the nervous system”. *Science*, 284(5411), pp. 96–98. [Cited on page 25.]

[Kubík, 2003] Kubík, A. (2003). “Toward a formalization of emergence.”. *Artificial life*, 9(1), pp. 41–65. [Cited on pages 3, 6, 38, and 41.]

[Ladyman et al., 2013] Ladyman, J.; Lambert, J.; and Wiesner, K. (2013). “What is a complex system?”. *European Journal for Philosophy of Science*, 3(1), pp. 33–67. [Cited on page 14.]

[Langton, 1990] Langton, C. G. (1990). “Computation at the edge of chaos: phase transitions and emergent computation”. *Physica D: Nonlinear Phenomena*, 42(1), pp. 12–37. [Cited on page 25.]

[Lavaei and Murray, 2012] Lavaei, J. and Murray, R. M. (2012). “Quantized Consensus by Means of Gossip Algorithm”. *IEEE Transactions on Automatic Control*, 57(1), pp. 19–32. [Cited on page 63.]

[LeCun et al., 1989] LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; and Jackel, L. D. (1989). “Backpropagation applied to handwritten zip code recognition”. *Neural computation*, 1(4), pp. 541–551. [Cited on page 27.]

[Ledoux, 1997] Ledoux, C. (1997). “An urban traffic flow model integrating neural networks”. *Transportation Research Part C: Emerging Technologies*, 5(5), pp. 287–300. [Cited on page 27.]

[Levin, 1998] Levin, S. A. (1998). “Ecosystems and the biosphere as complex adaptive systems”. *Ecosystems*, 1(5), pp. 431–436. [Cited on page 24.]

[Lewes, 1875] Lewes, G. H. (1875). “Problems of life and mind”. *London: Kegan Paul, Trench, Turbner, and Co*, 2. [Cited on pages 13, 15, and 18.]

[Lewis and Whitehead, 2011] Lewis, C. and Whitehead, J. (2011). “Repairing Games at Runtime or, How We Learned to Stop Worrying and Love Emergence”. *IEEE Software*, 28(5), pp. 53–59. [Cited on pages 42 and 43.]

[Liao and Chen, 2001] Liao, P.-Y. and Chen, J.-S. (2001). “Dynamic trading strategy learning model using learning classifier systems”. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 783–789. IEEE. [Cited on page 28.]

[Lopez, 2015] Lopez, F. (2015). “Modeling emergence of norms in multi-agent systems by applying tipping points ideas”. *arXiv preprint arXiv 1508.04531.* [Cited on page 30.]

[Macal and North, 2005] Macal, C. M. and North, M. J. (2005). “Tutorial on agent-based modeling and simulation”. In *Proceedings of the 37th conference on Winter simulation*, pages 2–15. Winter Simulation Conference. [Cited on page 29.]

[Macal and North, 2010] Macal, C. M. and North, M. J. (2010). “Tutorial on agent-based modelling and simulation”. *Journal of simulation*, 4(3), pp. 151–162. [Cited on page 28.]

[Macy and Willer, 2002] Macy, M. W. and Willer, R. (2002). "From factors to actors: Computational sociology and agent-based modeling". *Annual review of sociology*, pages 143–166. [Cited on page 29.]

[Mamei and Zambonelli, 2004] Mamei, M. and Zambonelli, F. (2004). "Self-organization in multi agent systems: A middleware approach". In *Engineering Self-Organising Systems*, pages 233–248. Springer. [Cited on page 62.]

[Martens et al., 2007] Martens, D.; De Backer, M.; Haesen, R.; Vanthienen, J.; Snoeck, M.; and Baesens, B. (2007). "Classification with ant colony optimization". *Evolutionary Computation, IEEE Transactions on*, 11(5), pp. 651–665. [Cited on page 62.]

[Maxwell et al., 2002] Maxwell, T. T.; Ertas, A.; and Tanik, M. M. (2002). "Harnessing complexity in design". *Journal of Integrated Design and Process Science*, 6(3), pp. 63–74. [Cited on page 5.]

[Miller and Page, 2009] Miller, J. H. and Page, S. E. (2009). *Complex adaptive systems: an introduction to computational models of social life: an introduction to computational models of social life.* Princeton university press. [Cited on pages 16, 25, and 26.]

[Miller and Bassler, 2001] Miller, M. B. and Bassler, B. L. (2001). "Quorum sensing in bacteria". *Annual Reviews in Microbiology*, 55(1), pp. 165–199. [Cited on page 62.]

[Minar et al., 1996] Minar, N.; Burkhart, R.; Langton, C.; and Askenazi, M. (1996). "The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations". [Cited on page 32.]

[Mnif and Muller-Schloer, 2006] Mnif, M. and Muller-Schloer, C. (2006). "Quantitative Emergence". In *2006 IEEE Mountain Workshop on Adaptive and Learning Systems*, pages 78–84. IEEE. [Cited on pages 6 and 34.]

[Mogul, 2006] Mogul, J. C. (2006). "Emergent (mis)behavior vs. complex software systems". *ACM SIGOPS Operating Systems Review*, 40(4), pp. 293. [Cited on pages 2, 15, and 47.]

[Montagna et al., 2012] Montagna, S.; Pianini, D.; and Viroli, M. (2012). "Gradient-Based Self-Organisation Patterns of Anticipative Adaptation". *2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems*, pages 169–174. [Cited on page 31.]

[Morris et al., 2014] Morris, A.; Ross, W.; Hosseini, H.; and Ulieru, M. (2014). "Modelling culture with complex, multi-dimensional, multi-agent systems". In *Perspectives on Culture and Agent-based Simulations*, pages 13–30. Springer. [Cited on page 30.]

[Moshirpour et al., 2012] Moshirpour, M.; Mousavi, A.; and Far, B. H. (2012). "Detecting Emergent Behavior in Distributed Systems Using Scenario-Based Specifications". *International Journal of Software Engineering and Knowledge Engineering*, 22(06), pp. 729–746. [Cited on pages 6, 39, and 41.]

[Moussaïd et al., 2011] Moussaïd, M.; Helbing, D.; and Theraulaz, G. (2011). "How simple rules determine pedestrian behavior and crowd disasters". *Proceedings of the National Academy of Sciences*, 108(17), pp. 6884–6888. [Cited on page 94.]

[Muller, 2004] Muller, J.-P. (2004). "Emergence of collective behaviour and problem solving". In *Engineering Societies in the Agents World IV*, pages 1—-20. Springer. [Cited on pages 19 and 52.]

[Müller-Schloer and Sick, 2008] Müller-Schloer, C. and Sick, B. (2008). "Controlled emergence and self-organization". In *Organic Computing*, pages 81–103. Springer. [Cited on page 16.]

[Nallur et al., 2015] Nallur, V.; Monteil, J.; Sammons, T.; Bouroche, M.; and Clarke, S. (2015). "Increasing Information in Socio-Technical MAS Considered Contentious.". In *Ninth IEEE Self-Adaptive and Self Organizing Workshops (SASOW)*. [Cited on page 136.]

[Nedos et al., 2006] Nedos, A.; Singh, K.; and Clarke, S. (2006). "Mobile ad hoc services: semantic service discovery in mobile ad hoc networks". In *Service-Oriented Computing–ICSOC 2006*, pages 90–103. Springer. [Cited on page 63.]

[Newman, 1996] Newman, D. V. (1996). "Emergence and strange attractors". *Philosophy of Science*, pages 245–261. [Cited on page 16.]

[Niazi and Hussain, 2011] Niazi, M. A. and Hussain, A. (2011). "Sensing Emergence in Complex Systems". *IEEE Sensors Journal*, 11(10), pp. 2479–2480. [Cited on pages 6, 25, 36, 37, 93, and 146.]

[Nicolis, 1989] Nicolis, G. (1989). "Physics of far-from-equilibrium systems and self-organisation". *The new physics*, 11, pp. 316–347. [Cited on page 16.]

[Noel and Zambonelli, 2014] Noel, V. and Zambonelli, F. (2014). "Engineering emergence in Multi-Agent Systems: Following the problem organisation". In *High Performance Computing & Simulation (HPCS), 2014 International Conference on*, pages 444–451. IEEE. [Cited on page 30.]

[Northrop et al., 2006] Northrop, L.; Feiler, P.; Gabriel, R. P.; Goodenough, J.; Linger, R.; Longstaff, T.; Kazman, R.; Klein, M.; Schmidt, D.; Sullivan, K.; and Wallnau, K. (2006). "Ultra-Large-Scale Systems - The Software Challenge of the Future". Technical report, Software Engineering Institute, Carnegie Mellon. [Cited on pages 1 and 4.]

[Olaru and Florea, 2009] Olaru, A. and Florea, A. M. (2009). "Emergence in Cognitive Multi-Agent Systems". In *CSCS17, the 17th International Conference on Control Systems and Computer Science, MASTS Workshop*, volume 2, pages 515–522. [Cited on pages 4 and 12.]

[Olfati-Saber, 2006] Olfati-Saber, R. (2006). "Flocking for multi-agent dynamic systems: Algorithms and theory". *Automatic Control, IEEE Transactions on*, 51(3), pp. 401–420. [Cited on page 30.]

[O'Toole et al., 2014] O'Toole, E.; Nallur, V.; and Clarke, S. (2014). "Towards Decentralised Detection of Emergence in Complex Adaptive Systems". In *2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems*, pages 60–69. IEEE. [Cited on pages 54 and 56.]

[Ottino, 2004] Ottino, J. M. (2004). "Engineering complex systems.". *Nature*, 427(6973), pp. 399. [Cited on page 2.]

[Page, 1954] Page, E. S. (1954). "Continuous inspection schemes". *Biometrika*, pages 100–115. [Cited on page 60.]

[Page, 2010] Page, S. E. (2010). *Diversity and complexity.* Princeton University Press. [Cited on pages 23 and 50.]

[Park et al., 1991] Park, D. C.; El-Sharkawi, M.; Marks, R.; Atlas, L.; Damborg, M.; et al. (1991). "Electric load forecasting using an artificial neural network". *Power Systems, IEEE Transactions on*, 6(2), pp. 442–449. [Cited on page 27.]

[Parks and Burrus, 1987] Parks, T. W. and Burrus, C. S. (1987). *Digital filter design.* Wiley-Interscience. [Cited on pages 113 and 152.]

[Parunak et al., 2002] Parunak, H. V. D.; Brueckner, S.; and Sauter, J. (2002). "Digital pheromone mechanisms for coordination of unmanned vehicles". In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 449–450. ACM. [Cited on page 62.]

[Paunovski et al., 2009] Paunovski, O.; Eleftherakis, G.; and Cowling, A. J. (2009). "Disciplined exploration of emergence using multi-agent simulation framework". *Computing and Informatics*, 28(3), pp. 369–391. [Cited on page 31.]

[Paunovski et al., 2008] Paunovski, O.; Eleftherakis, G.; and Cowling, T. (2008). "Framework for empirical exploration of emergence using multi-agent simulation". In *Workshop on Complex Systems Modelling and Simulation. Luniver Press*, pages 1–31. [Cited on pages 15 and 39.]

[Pearson, 1900] Pearson, K. (1900). "X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302), pp. 157–175. [Cited on page 68.]

[Poff et al., 1996] Poff, N. L.; Tokar, S.; and Johnson, P. (1996). "Stream hydrological and ecological responses to climate change assessed with an artificial neural network". *Limnology and Oceanography*, 41(5), pp. 857–863. [Cited on page 27.]

[Procházka and Olševičová, 2015] Procházka, J. and Olševičová, K. (2015). "Monitoring Lane Formation of Pedestrians: Emergence and Entropy". In *Intelligent Information and Database Systems*, pages 221–228. Springer. [Cited on pages 30, 32, 34, 83, 94, and 96.]

[Ramdane-Cherif, 2007] Ramdane-Cherif, A. (2007). "Toward Autonomic Computing: Adaptive Neural Network for Trajectory Planning.". *IJCINI*, 1(2), pp. 16–33. [Cited on page 27.]

[Randles et al., 2007] Randles, M.; Zhu, H.; and Taleb-Bendiab, A. (2007). "A Formal Approach to the Engineering of Emergence and its Recurrence". *Proc. of EEDAS-ICAC*, pages 1–10. [Cited on pages 6, 40, and 41.]

[Reynolds, 1987] Reynolds, C. W. (1987). "Flocks, herds and schools: A distributed behavioral model". *ACM Siggraph Computer Graphics*, 21(4), pp. 25–34. [Cited on pages 31 and 89.]

[Rocha, 1998] Rocha, L. M. (1998). “Selected self-organization and the semiotics of evolutionary systems”. In *Evolutionary systems*, pages 341–358. Springer. [Cited on page 22.]

[Ronald et al., 1999] Ronald, E. M. A.; Sipper, M.; and Capcarrère, M. S. (1999). “Design, Observation, Surprise! A Test of Emergence”. *Artificial Life*, 5(3), pp. 225–239. [Cited on pages 18 and 40.]

[Ross, 1997] Ross, W. D., editor (1997). *Aristotle: Metaphysics.* Sandpiper Books. [Cited on page 13.]

[Rupert et al., 2008] Rupert, M.; Rattrout, A.; and Hassas, S. (2008). “The web from a complex adaptive systems perspective”. *Journal of Computer and System Sciences*, 74(2), pp. 133–145. [Cited on page 25.]

[Russell et al., 1995] Russell, S.; Norvig, P.; and Intelligence, A. (1995). “A modern approach”. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25. [Cited on page 30.]

[Sapere, 2015] Sapere (2015). “Sapere-Project”. [Cited on page 31.]

[Sauter et al., 2005] Sauter, J. A.; Matthews, R.; Van Dyke Parunak, H.; and Brueckner, S. A. (2005). “Performance of digital pheromones for swarming vehicle control”. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 903–910. ACM. [Cited on page 62.]

[Sawyer, 2005] Sawyer, R. K. (2005). *Social emergence: Societies as complex systems.* Cambridge University Press. [Cited on page 12.]

[Seeley and Visscher, 2004] Seeley, T. and Visscher, P. K. (2004). “Group decision making in nest-site selection by honey bees”. *Apidologie*, 35(2), pp. 101–116. [Cited on page 62.]

[Seth, 2008] Seth, A. K. (2008). “Measuring emergence via nonlinear Granger causality.”. In *ALIFE*, volume 2008, pages 545–552. [Cited on pages 6, 35, 37, and 146.]

[Shalizi and Moore, 2003] Shalizi, C. R. and Moore, C. (2003). “What Is a Macrostate? Subjective Observations and Objective Dynamics”. *arXiv preprint cond-mat/0303625*, page 15. [Cited on page 35.]

[Shannon, 1948] Shannon, C. (1948). “A Mathematical Theory of Communication”. *Bell System Technical Journal*, 27, pp. 379–423, 623–656. [Cited on page 33.]

[Singh et al., 2013] Singh, V.; Singh, G.; and Pande, S. (2013). "Emergence, Self-Organization and Collective Intelligence – Modeling the Dynamics of Complex Collectives in Social and Organizational Settings". In *2013 UKSim 15th International Conference on Computer Modelling and Simulation*, pages 182–189. IEEE. [Cited on page 4.]

[Stephan, 1999] Stephan, A. (1999). "Varieties of Emergence". *Evolution & Cognition*, 5(1), pp. 49–59. [Cited on page 5.]

[Sycara, 1998] Sycara, K. P. (1998). "Multiagent systems". *AI magazine*, 19(2), pp. 79. [Cited on pages 29 and 50.]

[Szabo and Teo, 2015] Szabo, C. and Teo, Y. M. (2015). "Formalization of Weak Emergence in Multiagent Systems". *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 26(1), pp. 6. [Cited on pages 39 and 41.]

[Tang and Mao, 2014] Tang, M. and Mao, X. (2014). "Information Entropy-Based Metrics for Measuring Emergences in Artificial Societies". *Entropy*, 16(8), pp. 4583–4602. [Cited on page 83.]

[Tay and Jhavar, 2005] Tay, J. C. and Jhavar, A. (2005). "CAFISS". In *Proceedings of the 2005 ACM symposium on Applied computing - SAC '05*, page 158, New York, New York, USA. ACM Press. [Cited on page 28.]

[Taylor et al., 2014] Taylor, A.; Dusparic, I.; Galván-López, E.; Clarke, S.; and Cahill, V. (2014). "Accelerating Learning in multi-objective systems through Transfer Learning". In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 2298–2305. IEEE. [Cited on page 29.]

[Teo et al., 2013] Teo, Y. M.; Luong, B. L.; and Szabo, C. (2013). "Formalization of emergence in multi-agent systems". *Proceedings of the 2013 ACM SIGSIM conference on Principles of advanced discrete simulation - SIGSIM-PADS '13*, page 231. [Cited on pages 6, 33, 38, and 41.]

[Tesfatsion, 2003] Tesfatsion, L. (2003). "Agent-based computational economics: modeling economies as complex adaptive systems". *Information Sciences*, 149(4), pp. 262–268. [Cited on page 25.]

[Tibshirani, 1996] Tibshirani, R. (1996). "Regression shrinkage and selection via the lasso".

Journal of the Royal Statistical Society. Series B (Methodological), pages 267–288. [Cited on page 58.]

[Toth and Vigo, 2002] Toth, P. and Vigo, D. (2002). "Models, relaxations and exact approaches for the capacitated vehicle routing problem". *Discrete Applied Mathematics*, 123(1), pp. 487–512. [Cited on page 62.]

[Vizzari et al., 2013] Vizzari, G.; Manenti, L.; and Crociani, L. (2013). "Adaptive pedestrian behaviour for the preservation of group cohesion". *Complex Adaptive Systems Modeling*, 1(1), pp. 1–29. [Cited on page 25.]

[Waldrop, 1993] Waldrop, M. M. (1993). *Complexity: The emerging science at the edge of order and chaos*. Simon and Schuster. [Cited on page 17.]

[Wang, 2010] Wang, F.-Y. (2010). "Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications". *Intelligent Transportation Systems, IEEE Transactions on*, 11(3), pp. 630–638. [Cited on page 25.]

[Weiss, 1999] Weiss, G. (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press. [Cited on pages 29 and 50.]

[Whitehead and Ballard, 2014] Whitehead, S. D. and Ballard, D. H. (2014). "A Role for Anticipation in Reactive Systems that Learn". In *Proc. 6th Intl. Workshop on Machine Learning*, pages 354–357. [Cited on page 28.]

[Wilensky, 1998] Wilensky, U. (1998). "NetLogo Flocking model". [Cited on page 90.]

[Wilensky, 1999] Wilensky, U. (1999). "NetLogo (and NetLogo user manual)". *Center for Connected Learning and Computer-Based Modeling, Northwestern University. http://ccl. northwestern. edu/netlogo*. [Cited on pages 81 and 107.]

[Wold et al., 1987] Wold, S.; Esbensen, K.; and Geladi, P. (1987). "Principal component analysis". *Chemometrics and intelligent laboratory systems*, 2(1), pp. 37–52. [Cited on page 58.]

[Wolfram, 1984] Wolfram, S. (1984). "Universality and complexity in cellular automata". *Physica D: Nonlinear Phenomena*, 10(1), pp. 1–35. [Cited on page 25.]

[Wooldridge, 2009] Wooldridge, M. (2009). *An introduction to multiagent systems.* John Wiley & Sons. [Cited on pages 29 and 50.]

[Woźniak et al., 2014] Woźniak, M.; Graña, M.; and Corchado, E. (2014). "A survey of multiple classifier systems as hybrid systems". *Information Fusion*, 16, pp. 3–17. [Cited on page 28.]

[Xiang and Lee, 2008] Xiang, W. and Lee, H. (2008). "Ant colony intelligence in multi-agent dynamic manufacturing scheduling". *Engineering Applications of Artificial Intelligence*, 21(1), pp. 73–85. [Cited on page 30.]

[Zambonelli et al., 2011] Zambonelli, F.; Castelli, G.; Ferrari, L.; Mamei, M.; Rosi, A.; Marzo, G. D.; Risoldi, M.; Tchao, A.-E.; Dobson, S.; Stevenson, G.; Ye, J.; Nardini, E.; Omicini, A.; Montagna, S.; Viroli, M.; Ferscha, A.; Maschek, S.; and Wally, B. (2011). "Self-aware Pervasive Service Ecosystems". *Procedia Computer Science*, 7, pp. 197–199. [Cited on page 31.]

[Zambonelli et al., 2015] Zambonelli, F.; Omicini, A.; Anzengruber, B.; Castelli, G.; De Angelis, F. L.; Di Marzo Serugendo, G.; Dobson, S.; Fernandez-Marquez, J. L.; Ferscha, A.; Mamei, M.; Mariani, S.; Molesini, A.; Montagna, S.; Nieminen, J.; Pianini, D.; Risoldi, M.; Rosi, A.; Stevenson, G.; Viroli, M.; and Ye, J. (2015). "Developing pervasive multi-agent systems with nature-inspired coordination". *Pervasive and Mobile Computing*, 17, pp. 236–252. [Cited on pages 30 and 31.]

[Zhu, 2001] Zhu, H. (2001). "SLABS: a formal specification language for agent-based systems". *International Journal of Software Engineering and Knowledge Engineering*, 11(05), pp. 529–558. [Cited on page 40.]