Large Scale Agent Interactions: Mathematical Modelling and Simulation

by
Yu Wang
B.Sc(Hons), M.Sc

A Thesis submitted in fulfilment of requirements for the degree of
Doctor of Philosophy of University of London and
Diploma of Imperial College

Intelligent Systems & Networks Group
Department of Electrical and Electronic Engineering
Imperial College London
University of London
2006

IN MEMORY OF DP

Abstract

One of the classical problems in modelling multi-agent systems is the computational complexity of the system increases exponentially with respect to the number of agents that are being modelled. Traditional techniques, such as and non-linear differential equations and discrete event simulation model agents individually. Hence they are of high computational complexity and difficult to solve (i.e. prove the existence and uniqueness of solutions).

In this thesis, we describe an established mapping between the G-Network parameters and agent simulations and propose a stochastic approach that provides performance estimates of systems at low computational cost. We model agents of the same nature (e.g. behavior) collectively and local interactions only when these are relevant, and allow hierarchical representation via the behavior of groups. In doing so, we separate out the link between the computational complexity of a model and the number of agents, and associate the computational complexity with the agent types (number of groups) and the number of locations that are being modelled.

The approach is designed to replace some traditional agent simulations so that we can avoid conducting many simulation runs when gathering the statistics and reduce computational complexity. It models the systems at different levels of detail and abstraction and is also able to model the systems when only insufficient information is available. However, it is not possible to model all systems (e.g. finding unique solutions); however the use of G-networks insures existence and uniqueness of solutions. We limit ourselves to time invariant systems where the parameters do not depend on time, although the system varies with time. The system of equations may be very large (i.e. infinite), we therefore limit our analysis to seeking steady-state solutions.

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to my supervisor, Prof. Erol Gelenbe, for his invaluable guidance, enthusiasm, continuous encouragement and support during my Ph.D. I have and will always benefit from the things he has showed me. I am grateful that he inspired me and taught me how to be organized and disciplined. It was not only me but also Erol who have spent a lot of time and effort on this work. Without his feedback and encouragement, it would not have been possible to finish this thesis. Special thanks go to my previous supervisor from Edinburgh, Dr. Jane Hillston, for her encouragement and friendly support.

I am grateful to my parents for their selfless and never-ending love, support and trust. Without them, it would not have been possible for me to keep challenging myself and pursuing new goals. I would like to thank my all friends, especially Andrew, Dai and Varol for their whole-hearted support during very hard times. Finally, I would like to thank a very, very special person, *DP*, who inspired me, showed me the world and helped me growing up. His determination and dedication make him a great role model. I dedicate this thesis to him for giving me the happiest, the saddest, and the most meaningful time in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The world that we live in is filled with large scale agent systems, in areas such as biology, ecology, finance, and transportation. They have been traditionally characterized by a large number of variables, nonlinearities and uncertainties. Examples of large scale systems are the human body in a biological context, where different organs made up of billions of cells work together towards the operation of the human body, or a computer network where devices like routers, servers, computers and printers co-operate to ensure the smooth running of a network. The complexity and diversity of such systems make them difficult to describe and model, and it is even harder to provide fast numerical predictions of the underlying processes of such systems.

As discussed in Chapter 2, there have been many attempts to model large scale agent systems, through the use of non-linear differential equations and discrete event simulations [9, 15, 19, 60, 70, 75, 83]. These traditional approaches have not been very successful due to the high computational complexity resulted by the the fact that agents are modelled separately. A well-known problem in modelling large scale systems is exponential explosion [66, 89], which refers to the fact that the computational complexity of the system increases exponentially with respect to the number of entities that are being modelled. In the field of cancer treatments for example, a drug's performance may be measured by the percentage of cancerous cells that remain in the body. However,

the population size of the cells makes it computationally infeasible to model them individually. In areas such as military planning and cancer treatment, it may also be operationally infeasible and expensive to carry out experiments. For example, during wartime, it is much more reasonable and practical to use modelling techniques rather than experiments to devise an optimal strategy for battlefield maneuvers.

To overcome the shortcomings of the traditional techniques, we proposed a novel stochastic approach, based on an extension to existing queuing networks, specifically G-networks [24–28, 31–42, 48, 55], that models large scale agent system and provides performance estimates of such systems at low computational cost. By modelling agents of the same nature (e.g. type or behavior) collectively, the approach eliminates the dependency that the computational complexity of a system has on the number of entities that are being modelled. Instead, the computational complexity is associated with the number of locations (e.g. the geographical locations) and the number of classes (e.g. type of agents) in the system (See Chapter 4). With the approach, we provide insight into agent systems in terms of their performance and behavior, identify the parameters which strongly influence them, and evaluate how well individual goals of specific agents are achieved. By using the mathematical models to compare the impact of alternatives, we provide the possibility of choosing plans and sets of parameters that best address the users' requirements.

Depending on the number of the agents that are of interest, the approach models the systems at different levels of detail. According to their characteristics, we categorize the agents as either *indistinguishable* or *distinguishable*. The term *indistinguishable agents* describes those that exist in large numbers, behave identically and have trivial individual impact on the system. In the field of biology, cancerous cells are a good example of indistinguishable agents. Because the effect of a single cancerous cell is negligible, we are more interested in aspects such as the density. On the other hand, there are systems contain a small number of agents, referred to as *distinguishable agents*, which may have similar or different behavior but more importantly which have noticeable impact

on the system on their own. These type of systems are common in the field of strategic military planning, where we study the performance of a particular agent or agent team.

In Chapter 3, we demonstrate how to model open systems that contain *indistinguishable agents*. With examples that are set in the field of biology, we illustrate how to construct models to represent agent interactions such as collaboration and competition. We also show how the mathematical models can be solved both numerically and analytically. By comparing the systems' outcomes under different parameter settings, the mathematical model helps us to identify the parameters that affect the system most as well as the trade-offs, if they exist. In addition, we can obtain the ranges of the parameters where the model has valid solutions. Work in this chapter was published in [45, 49].

In Chapter 4, we study systems containing a large number of geographical locations, fewer but more influential agents in the context of urban military planning. We discuss how to construct models for scenarios where the outcomes are determined by the behavior of a particular agent or agent team. Similar to the previous chapter, we demonstrate that the model helps us identify the parameters that impact the system most as well as setting under which agents or agent teams operate in the most cost-effective manner. Experiments show that the proposed approach is capable of modelling systems at different levels of abstraction: we obtain an estimated outcome of scenarios set in a large space by modelling them in a much smaller one. In doing so, the computational complexity is reduced further.

Also in Chapter 4, we discuss how the mathematical approach is validated against a simulator [43, 44]. By modelling the same scenarios, we compare the obtained results and examine the discrepancies. Designed with different objectives in mind, the simulator offers a visual representation of the underlying processes of a system whereas the mathematical approach examines the system at a higher level of abstraction and provides an estimated numerical outcome of the system. Despite the differences, both approaches can be used to gain insights into the systems. The differences of the two approaches cause minor

discrepancies in the results, which will be discussed in detail. Work in this chapter was published in [46, 50, 92, 93]. The following table lists the differences between the systems that are studied in Chapter 3 and 4 (See Table 1.1).

Table 1.1: Indistinguishable systems *Vs.* distinguishable systems

Indistinguishable systems	distinguishable systems
E.g. Biological Systems	E.g. Urban Environment
Agents (E.g. cancerous cells) exist in large quantities	Agents (E.g. police) exist in small quantities, the population (civilians) is more numerous
Behave in a similar manner with statistical variations at the individual level	Similar or different behavior
Trivial individual impact on the system only the collective impact is noticeable	Noticeable individual impact on the system for police, less for civilians
Unnecessary and impractical to model separately	Model such agents separately
Interested in their density	Interested in the performance of an individual agent or more generally of an agent group

Initially, the approach is designed under the assumption that information about the interactions within the system is available. This is not always the case for large scale agent systems that are known for their uncertainties. To overcome such situations, we extend the approach by integrating the historical observation data obtained from simulations, into the mathematical model. Experiments show that the extended approach is able to derive the agent interactions and average agent behaviors from multiple simulation runs, fill in the information gap, and provide estimated outcomes of events happening. Comparing with the cases where we have complete information of the system, lack of information affects the precision of the results. However, the obtained performance estimate is, by all means, a good reference when only insufficient information is available. In Chapter 5, in addition to describing this extension, we also discuss the potential applicability of the observation method to the field of goal recognition. Work in this chapter was published in [51]. We conclude the thesis with a discussion of the research contribution, and potential extensions to the approach in Chapter 6.

Chapter 2

Background

In this chapter, we present background information on agents, queueing theory, mathematical modelling, biology systems, and related work.

2.1 Agents

2.1.1 What is an Agent?

Even though the word 'agent' has been used for nearly two decades, there is still not a succinct definition that includes all of the things that most researchers and developers consider agents to be. Dated back to the early days of Artificial Intelligence, Hewitt [57] proposed an 'actor', which is a self-contained, interactive and concurrently executing object. Being the early model of an agent, an actor has some encapsulated internal states and responds to messages from other similar objects.

Depending on the application field, researchers offered a variety of interpretations of 'agent'. Russell and Norvig [85] defined a particular agent, the AIMA agent (short for Artificial Intelligence: a Modern Approach), to be anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. Maes [76] interpreted autonomous agents to be computational systems that inhabit some complex dynamic environment, sense and act autonomously in it, and

accomplish a set of goals or tasks that they are designed for. Smith, Sypher and Spohrer [88] proposed the KidSim agent to be a persistent software entity dedicated to a specific purpose. Hayes-Roth [53] perceived intelligent agents as entities that continuously perform perception of dynamic conditions in the environment, affect conditions and reasons to interpret perceptions, solve problems, draw inferences and determine actions. Brustoloni [14] and Franklin [29] stated that "Autonomous agents are systems capable of autonomous, purposeful action in the real world".

Other definitions can be found in [7, 30, 82, 87, 91, 97]. We believe that majority of researchers agree with the way Wooldridge and Jennings [96] defined agents. They perceive agents as computer systems that are capable of taking autonomous action in the situated environment to meet their design objectives. Agents are able to act without direct human intervention, and have control over their own actions and internal states. This agent definition is what will be used in this thesis.

Nwana and Ndumu [80] split the agent related research into two generations, dating from 1977 to 1990, and 1990 to the present. The work in the first period concentrated mainly on deliberative-type agents with symbolic internal models. The main issues that are focused on are the interaction and communication between agents, the decomposition and distribution of tasks, coordination and cooperation, and conflict resolution via negotiation. The current generation emphasizes agent theories, architectures and languages and significantly broadens the agent typology.

2.1.2 Agent Classifications

One of the most popular ways to classify agents is by the properties that they exhibit. The definitions suggest that agents are reactive, autonomous, goal-oriented, temporally continuous, communicative, capable of learning, mobile, and fiexible. Other properties such as inferential capability, personality, etc. are mentioned in [13, 21, 30]. Agents can be classified as either static or mobile according to their ability to move. They can also

be referred to as deliberative or reactive, collaborative, interface agents and smart agents. We can classify agents according to the tasks they perform, their control structure, the range and sensitivity of their senses, the range and effectiveness of their actions, and the number of internal states they possess. In this thesis, our agents are reactive, autonomous, goal-oriented, and mobile.

Jennings and Wooldridge [64] and Nwana [80] propose a common classification by collapsing the multi-dimensional space that agents exist in, and identifying seven types of agents. The types are: collaborative agents, interface agents, mobile agents, information/internet agents, reactive agents, hybrid agents and smart agents. Other classification means can be found in [13,14,30,82]. Agents can belong to multiple categories, and their applications often combine multiple categories.

2.1.3 Multi-agent System

According to Weiss [95], the term *multi-agent system* (or MAS) refers to systems comprising an environment, objects, agents, relations, operations and operators. An environment, E, is a place that generally has a volume with objects O situated in it. At any given moment, it is possible to associate any object with a position in E. They are passive and can be perceived, created, destroyed and modified by the agents. Agents A are specific objects, representing active entities of the system. Relations, R, directly link objects (hence indirectly link agents) to each other. An assembly of operations, Op, allows agents to perceive, produce, consume, transform and manipulate objects. Operators represent the application of operations and the reaction of the world.

MAS, and the systems we are interested in are similar. The concept of actions, fundamental to multi-agent systems, is based on the fact that the agents carry out actions which are going to modify their environment, and thus their future decisions. Similarly, the distinguishable agents that are studied in Chapter 4 exist in the system and take actions to reach a location or attack an enemy. These actions also have impact on the

global environment. We therefore consider large scale systems as MAS, and use agents to represent the individuals in the system.

2.1.4 Existing Approaches that Model Multi-agent Systems

So far, we have briefly introduced the definition and classification of the agents. In this section, we discuss how researchers in the field model multi-agent systems.

Amin and Mikler [9] proposed an agent-based approach to Distance Vector Routing. In it, route discovery is manifested in the movement of agents carrying routing information from one node to another, rather than the propagation of individual update messages. The number of agents is manipulated based on the resource availability. They stated that individual agents do not have a global view of the system so a high degree of coordination is required. To address this matter, they exploited an agent's stigmergetic property. Pheromones which aid the agents in population control are referred to as Node Pheromones, which are expressed by the degree of volatility of the pheromone and the time in an equation. When the Node Pheromone is less than a Termination Threshold, the agents kill themselves. Similar to one of the potential applications of our work, they use the mathematical modelling technique to model agent populations in an unpredictable environment. In their work, the agent itself controls the alive or death status, and the local decision indirectly affects the overall population. In the systems studied in this thesis, the alive/death status of the agents are only controlled by the actions of the others.

Haynes and Wainwright [54] proposed a genetic programming (GP) [72] technique to evolve programs to make autonomous agents capable of learning how to survive in a hostile environment by providing a set of terminals and functions. The idea is that when a random population of programs is constructed, they evolve into successive generations hopefully with a better average fitness. The fitness function of the GP is an interpreter which maps these programs into actions in the minefield. After a certain number of generations, the individual program with the best fitness is selected to be the solution.

As a companion, a simulator is built to guide agents through mine-fields. It randomly generates a new environment in which to judge each generation.

There are other ways to model agent systems, as listed in the Agent-Oriented Methodologies survey [62]. Burmeister [15] defined three models for analyzing an agent system, the agent model, the organizational model and the cooperation model. In the agent model, agents and their environments are identified using an extension of Class-Responsibility-collaborator cards for including beliefs, motivations, plans and cooperation attitudes. The organizational model proposes the identification of agents' roles and the elaboration of diagrams by using Object Modelling Technique [84] notation for the inheritance hierarchy and the relationships between agents. The cooperation model identifies the cooperations and cooperation partners, the types of interchanged messages and analyzes protocols. Burmeister models the system from an abstract perspective, and focuses on agents' internal structures and cooperations. According to Burmeister's approach, we model the system at the cooperation level, for we are mainly interested in the interactions between the individuals. Unlike our approach, his approach does not provide information about the outcome of each option.

Kinny etc. [70] proposed an Agent Modelling Technique for systems of Belief, Desire and Intention (BDI) agents. The two main levels are defined for modelling such systems are external and internal level. The former decomposes the system into agents and the definition of their interactions, whereas the latter models the BDI agents through the belief model, the goal model and the plan model. These models describe the beliefs about the environment, the goals and events that an agent can adopt or respond to, and the plans that can be used to achieve the goals. The plans of achieving a goal are first analyzed, then represented in graphical notations. Agents' beliefs on the environment are modelled and represented notationally. Differing from our approach, their technique is based on notation. Similar to Burmeister's approach, it focuses on an agent's internal structure and the cooperation between agents. It reveals the relationships within a system but does not provide any information about the

quantity of the agents, and is therefore unable to compare the outcome of different options.

Cysnerios and Yu [19] suggested a requirements engineering methodology which takes into account an agent's characteristics such as autonomy, intentionality and sociality. They build a lexicon, which is used as a front-end for model construction. The methodology notationally represents and models social structures such as relationships and social roles of large scale systems. Similar to the others, it does not reveal the quantity of individuals or compare alternative options. However, model validation and verification are easy to carry out. The agent-oriented technique reveals conflicting views between agents, divergent goals, different ways of accomplishing goals etc.. The viewpoint technique uncovers how different actors perceive processes and relationships, and achieve goals. Given an objective, their model can self-propose options, whereas in our approach, all the options are provided by the modeler.

Huang *et al* [60] proposed an approach to analyze large electric power flow. They exploit the degree of "criticality" of the power system by constructing a M-level model, and associating algorithmic arrangement as one unit. This approach eliminates the process whereby engineers supply models and problem statements and experts design algorithms to solve them numerically. It utilizes aggregation/disaggregation techniques for large scale computational problems. The solution algorithm is executed in parallel from node to node of a set of ith level criticality with a detailed model on the critical portion and an aggregated model describing the less critical part. The solution is then passed to the $(i + 1)$th level, disaggregated and treated as the initial estimate for this level. The aggregation/disaggregation procedures can only run top-down or iteratively and the procedure itself is independent of the numerical algorithm at each level.

The solutions passing between different levels are available continually and their accuracy improves as the process goes on. Eventually the approximation solution converges to the exact solution. In traditional approaches, solutions are only available when the overall computation is completed. This approach is therefore preferable when the speed of

obtaining a solution is critical. We use fixed point iteration with randomly assigned values to calculate queues' traffic intensities. The fixed point iteration stops when the traffic intensities are within a reasonable range of their accurate value. Similar to their approach, we are able to provide estimated values for the parameters during the simulation.

2.2 Queueing Theory

2.2.1 A Brief History

According to Hlynka [58], queueing theory dates back to primitive man, for an early queue is described in the Bible. Hlynka, Cooper [17] and the others believe that the work [20] of A. K. Erlang [5] of Copenhagen Telephone Company marks the starting of the mathematical queueing theory [12, 18, 22, 23, 56, 61, 78, 90, 98]. In the early 1900s, Erlang studied the problem of telephone networks, during which he worked out a formula to calculate the probability that a customer has to wait for an available line when attempting to call outside the village. His simple but important formulas of tele-traffic engineering inspired many people in the area.

According to [6], the terms "queueing system" and "a complex queueing problem" were first used in 1951 by Kendall [68] in the Journal of the Royal Statistical Society, and "queueing theory" first occurred in 1954 in Science News. In 1952, Lindley proposed a way to set up integral equations to describe certain queueing systems, known as "Lindley's Integral equation" [74]. Kendall then introduced the A/B/C type queueing notation in 1953. Gani and Prabhu proposed the theory of continuous time storage models during 1956 – 1963. Frank Haight then introduced the concepts of balking, reneging and parallel queues in 1958. In the same year, H. White and Christie were the first to consider server breakdown. The use of queueing for computer performance evaluation began around 1970. During the last decade, Gelenbe introduced negative customers and signals [34, 36, 39, 40, 47, 48] into the traditional queueing network. The in-depth history of queueing theory can be found in [65, 86] and in a state of the art survey [10].

Queueing theory provides a mathematical basis which aims to understand and predict the behavior of communication networks. From humble beginnings, Queueing Theory now spans such diverse areas as telecommunications, computer science, manufacturing, air traffic control, military logistics, biology, and theme parks design.

2.2.2 A Queueing System

A queue is a common phenomena that occurs frequently in our daily routine, for instance, supermarket checkouts, doctor's waiting rooms, etc.. Kleinrock [71] defined queueing as the phenomena of "standing, waiting, and serving". He also suggested that "Any system in which arrivals place demands upon a finite capacity resource may be termed a queueing system". A *queueing model* is an abstract description of such a system, where the system's physical configuration is represented by servers and their interactions. It represents the stochastic nature of the arrival and service process. Figure 2.1 is an example; it describes a system with a single server, has external customer arrivals and departures from the system.

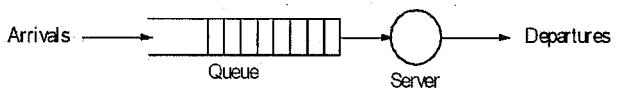


Figure 2.1: A simple system

The Terminology and Components of Queueing Theory

The five components of a queueing system are: inter-arrival time distribution; service-time distribution; number of servers; queueing discipline; and number of buffers. These are explained below.

The arrival and service process can be classified as *deterministic* or *probabilistic.* When the process is deterministic, the number of customers arriving or leaving the system per time unit is fixed. If the process is probabilistic, it may either depend on other inter-arrival times and/or the service process, or consist of independent and identically distributed inter-arrival times (e.g Markovian process). The Markov process bears the

name of its inventor A. A. Markov [77] who first defined and investigated the process in 1907. According to Markov, a set of random variables X_n forms a Markov chain if the probability that the next value (state) is $x_{(n+1)}$ depends only upon the current value (state) x_n and not upon any previous values. This characteristic of the process is often referred to as *memoryless.*

A popular assumptions in the queueing theory is that the customers enter and leave the system in a *Poisson process.* This is due to the fact that the Poisson process has a nice Markov property, where the conditional probability distribution of future states of the process, given the present state and all past states, depends only upon the current state and not on any past states. It is assumed that time is divided into small intervals of length Δt. The occurrence of an arrival in one interval is independent of the others. Let $\Delta t \rightarrow 0$; then the arrivals are said to follow a Poisson process. One of the properties of the Poisson process is that the time between arrivals is exponentially distributed, i.e., its distribution function $F_x(t)$ is given by $F_x(t) = 1-e^{-\lambda t}$.

The queueing service discipline describes the order in which customers are taken from the queue. The common service disciplines are:

1. First come first serve (FCFS)
2. Job priority
3. Shortest processing time first
4. Largest processing time first

When the arriving customers are distinguishable according to groups, the priority among groups may be established.

The common notation of a queue is $A/B/m$ or $A/B/m/K$, where m is the number of servers. K represents the buffer size and it is often omitted in the case of an infinite buffer size. A queue has either *limited* or *unlimited* capacity to hold its customers which

is generally determined by its buffer's available physical space. When the buffer is full, the newly arriving customers are blocked or lost. A and B are the arrival and service processes, and they are chosen from one of the following: M - Markov (exponential distribution); D - Deterministic (all customers have the same value); or G - General (arbitrary distribution). A common queue is the $M/M/1$ queue, in which both inter-arrival time and service time are exponentially distributed with rate λ and μ. It denotes a system with has only one server, infinite buffer and with FCFS as the service discipline.

Other types of queues such as $M/D/1$, its service time is deterministic (fixed regardless of number of customers). The $M/G/1$ type denotes whose arrival rate is exponential and the service rate has a general or arbitrary probability distribution. Further details on the different types of queues may be found in [47, 56, 71].

State Transition Diagram

We often use the number of customers in the system to denote the state of the system. The set of possible values (or states) that a system may take is called its state space. A state transition diagram, sometimes referred to as a transition rate diagram, is a graphical representation of the transition between the states of a process. The arcs denote the one-step transitions between states. For instance, in Figure 2.2, the transitions out of state $k-1$ of the $M/M/1/K$ queue lead either to state k at rate λ (arrival) or to state $k-2$ at rate μ (departure). The queue has a buffer size of K, and customers arriving to a full buffer are lost.

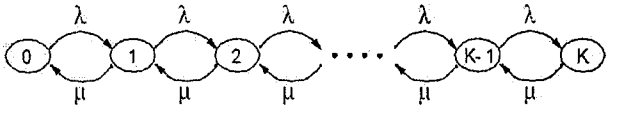


Figure 2.2: The state transition diagram of an M/M/1/K queue

Equilibrium

When a system reaches a stage that the probability of being in a certain state does

not depend on the time, we say that the system has reached steady state or equilibrium. In the steady state, the incoming customer flow equals the outgoing flow. We use $p(k)$ to present the probability that the system is in state k. Examining the state $k-1$ in equilibrium, we observe that the flow rate into it is $\lambda p_{k-2} + \mu p_k$, and the flow rate out is $(\lambda+\mu)p_{k-1}$. In equilibrium, the incoming flow and the outgoing flow must be the same so we have $\lambda p_{k-2} + \mu p_k = (\lambda + \mu)p_{k-1}$. During its lifetime, a system goes through transient states until it reaches steady states. However, the steady state does not always exist or may not be reached before the system ends.

One of the parameters that we are interested in is the server's utilization ρ (traffic intensity). It represents the fraction of time, on average, that a server is busy. It is the ratio of the rates that customers enter the system (arrival rate λ) to the maximum rate at which the system can perform this work (service rate μ). Knowing the utilization, we can work out other aspects of the queue, such as the average queue length. More details about utilization can be found in [47, 71].

Network of Queues

In reality, customers often require several services that are provided by different servers before they leave the system. Such a system is composed of a number of queues, and referred to as a *network of queues*. The number of servers in a queueing network corresponds to the number of operations that the system offers.

We classify queueing networks as *open* [10], *closed* [81] or *mixed*. In an open network, customers enter from outside, receive service at one or more servers and eventually leave the network. In a closed model, the number of customers flowing in the system is fixed so for each external departure there is an external arrival. A mixed network is one that is composed of both open and closed subnetworks. In this thesis, we study how to model open networks and close networks in Chapter 3 and 4 respectively.

2.2.3 G-Networks

Inspired by the excitation and inhibition behaviors in neural networks [32, 33, 35, 38], Gelenbe proposed a new class of queueing network in 1991. It is often referred to as G-Networks [34, 41], and was derived during the QMIPS [63] project. Differing from traditional queueing networks, customers in G-Networks are either *negative* or *positive*. Positive customers behave the same as those in the traditional networks, their arrival contributes to the queue length and they receive service in the usual manner. On the other hand, negative customers neither contribute to the queue length nor receive service. An arrival of a negative customer randomly picks and removes a positive customer when the queue is not empty; otherwise, it has no effect and itself is discarded from the system. Negative customers can be viewed as means to remove or destroy unwanted work load from the system. The negative and positive customers can either arrive from outside the network or move from one server to another. Customers leaving one queue for another can either remain positive or become negative.

The nodes in G-Networks obey the flow conservation rule, which states that for each node, the number of its incoming customers equals to the number of outgoing customers in the steady state. We use system equations (stationary Chapman-Kolmogorov equations) [25, 47, 71] to denote this. Using a network with n queues as an example, the service rates at each queue are $r(1), r(2), ..., r(n)$. We use the vector $k = (k_1, k_2, ..., k_n)$ to represent the state of the network, where k_i is the length of queue i. The vector $k + e_{im}$ is denoted as $(k_1, ..., k_i + m, ..., k_n)$. Similarly $k + e_i + e_{jm} = (k_1, ..., k_i + 1, ..., k_j + m, ..., k_n)$, $k - e_i = (k_1, ..., k_i - 1, ..., k_n)$, and $k + e_i - e_j = (k_1, ..., k_i + 1, ..., k_j - 1, ..., k_n)$.

Λ_i and λ_i are used to denote the positive and negative customers arrive at server i from outside the network. After the service, a customer leaves queue i for queue j in a positive form with probability $p^+(i,j)$ or in a negative form with probability $p^-(i,j)$, or departs from the network with probability $d(i)$. $p(k)$ denotes the stationary probability distribution $p(k) = \lim_{t\to\infty} Pr[k(t) = k]$. The following equation of a G-Network model

describes all the possible moves from neighboring states to state k:

$$
\begin{aligned}
& p(k)\sum_{i=1}^{n}[\Lambda(i)+(\lambda(i)+r(i))1[k_i>0]] \\
= & \sum_{i=1}^{n}(p(k+e_i)r(i)d(i)+p(k-e_i)\Lambda(i)1[k_i>0]+ \\
& \lambda(i)\sum_{m=1}^{\infty}\pi_{im}p(k+e_{im})+\sum_{j=1}^{n}(p(k+e_i-e_j)r(i)p^{+}(i,j)1[k_j>0]+ \\
& p(k+e_i)r(i)p^{-}(i,j)1[k_j=0]+\sum_{m=1}^{\infty}\pi_{jm}p(k+e_i+e_{jm})r(i)p^{-}(i,j)+ \\
& \lambda(i)\sum_{m=1}^{\infty}\pi_{im}\sum_{s=0}^{m-1}p(k+e_{is})1[k_i=0]+ \\
& \sum_{m=1}^{\infty}\pi_{jm}\sum_{s=1}^{\infty}p(k+e_i+e_{js})r(i)p^{-}(i,j)1[k_k=0])).
\end{aligned}
\tag{2.1}
$$

Each term represents a transition from a neighboring state to the current one. The term $\sum_{i=1}\lambda(i)\sum_{m=1}^{\infty}\pi_{im}\,p(k+e_{im})$ states that negative customers arrive at queue i and remove e_{im} positive customers. The term $\sum_{j=1}^{n}(p(k+e_i-e_j)\,r(i)p^{+}(i,j)\,1[k_j>0]$ corresponds to the situation that, after receiving services at queue i, the customers join queue j in the positive form. The term $1[k_j>0]$ represents a function that takes the value of 1 when $k_j>0$, and 0 otherwise. In Chapter 3, we show how to derive the system equations for a describe system.

Often we are interested in certain statistics in a network, for example, the traffic intensity, i.e. the availability of devices. Traffic intensities are solutions to the system equations as explained in [47]. The traffic intensity of queue i is as below:

$$
\rho_i=\frac{\sum_j\rho_j r(j)p^{+}(j,i)+\Lambda_i}{r(i)+\sum_j\rho_j r(j)p^{-}(j,i)+\lambda_i} \tag{2.2}
$$

The term $\sum_j\rho_j r(j)p^{+}(j,i)$ corresponds to the positive customers arriving in queue i after receiving service in other queues. Comparing the obtained traffic intensity (2.2) with the traditional form, we observe that $\sum_j\rho_j r(j)p^{-}(j,i)+\lambda_i$ positive customers are removed from queue i under the effect of incoming negative customers.

Gelenbe demonstrated that this model, given exponential service time, Poisson external arrivals, the usual independence assumptions for service times, and Markovian customer movements between queues, has product form when the unique solution exists to the non-linear equation (See Equation 2.2) such that $\rho_i < 1$. The product form of $p(k)$ is therefore as below:

$$p(k) = \prod_{i=1}^{n} [1 - \rho_i] \rho_i^{k_i} \tag{2.3}$$

The result of the product form has been generalized to the case where a negative customer can destroy a batch of positive customers in [27,37,47]. In this case, a negative customer arrives, and reduces the queue length by some integer random variable B_i. If there are less positive customers in the queue, it will be reduced to zero and the negative customer itself will simple disappear. The batch removal size distribution at queue i can be given by some arbitrary $P[B_i = m] = \pi_{im}, m \geq 1$. A function $f_i(x) = [1 - \sum_{m=1}^{\infty} \pi_{im} x^m]/[1 - x]$ was introduced to represent the average size of the batch. The traffic intensity can therefore be represented as below:

$$\rho_i = \frac{\sum_j \rho_j r(j) p^+(j,i) + \Lambda_i}{r(i) + \sum_j \rho_j r(j) p^-(j,i) + \lambda_i f_i(\rho_i)} \tag{2.4}$$

In a later work [36], Gelenbe proposed another customer type, the *signal*. On arrival, a signal forces a customer to move instantaneously to another queue according to a Markovian routing rule, or to leave the network. He pointed out that the triggered motion is useful in representing effects such as: tokens in Petri nets; in modelling systems where both customers and work can be instantaneously moved from one queue to an other upon certain events; and also for certain behaviors encountered in parallel computer system modelling. Signals do not discard positive customers, but instead move them to another queue in the system. A signal has no effect on an empty queue and simply disappears.

In a network of n queues, customers arrive from outside the system according to a

Poisson process of rate $\Lambda(i)$ and signals arrive with rate $\lambda(i)$. After being serviced at queue i, a customer may leave for queue j as a customer with probability $p^+(i,j)$, or as a signal with probability $p^-(i,j)$ or leave the system with probability $d(i)$. A signal arrives at a non-empty queue with probability $q(i,j)$, and triggers the instantaneous passage of a customer from queue i to some other queue j. The traffic intensity of queue i can be expressed as:

$$\rho_i \equiv \frac{\sum_j \rho_j [r(j)p^+(j,i) + \lambda^-(j)q(j,i)] + \Lambda(i)}{r(j) + \sum_j r(j)\rho_j p^-(j,i) + \lambda(i)} \tag{2.5}$$

Comparing the traffic intensities (See Equation 2.2) and (See Equation 2.5), we observe that negative customers and signals *both* move positive customers out of the queue. The special characteristic of G-Networks with signals, being able to trigger customers moving between queues, leads to numerous potential application areas such as *load balancing* and *data networks*:

Load balancing [1, 2, 8] has become one of the major issues relate to real-time management. It was first introduced to compensate for the fact that networks are faster than the systems they connect. Its first generation was therefore to ensure HTTP sessions are distributed among several IP hosts. When one server is overloaded, some of its incoming requests are sent to others to ensure that overall load is well-balanced across all available resources. Signals can be viewed as an explicit representation to replace work from highly utilized to less utilized queues. G-Networks can be used to model large computer systems and networks in which signaling functions such as negative customers and triggers are used to achieve flow and congestion control.

Gelenbe proved the existence of solutions for the non-linear traffic equations and established the product form of the G-Networks with signals. The G-Network are then extended to include multiple classes of positive and negative customers [26, 39], multiple classes with queue flushing [28] and state dependent batch removal [47, 55]. The existence and uniqueness of the product form solution and related issues have been discussed

in [34, 36, 48]. Other aspects of G-Networks, such as stability and stationarity, are discussed in [11,24,28,42].

Another major extension of G-Networks is the *reset* [40] customer type. Arriving at a non-empty queue, a reset modifies the queue length to a random number whose distribution is identical to the stationary distribution of the queue. A reset has no effect on a non-empty queue and is immediately discarded. Gelenbe identified the existence of the product form solution under the usual assumptions, and pointed out that even though the balance equation of such model is significantly different from the others, it is still identical to an ordinary G-Network with a large positive customers arrival rate.

2.3 Mathematical Modelling

Mathematics [94], like many other languages that we speak, be it French or English, is a language that we use to express ourselves, to communicate and to manipulate ideas, especially those about the surrounding world. It is widely agreed that mathematical areas like statistics, optimization, probability, queueing theory, modelling, etc., are essential for making difficult choices in public policy, health, business, finance and many other human endeavors [4]. Mathematics is said to be at the heart of a multitude of decisions, including those that generate electric power economically, making a profit in financial markets, approving effective new drugs, flying aircraft safely, managing complex construction projects, and choosing new business strategies.

Depending on the amount of the prior available information, models are classified as either *white-box* or *black-box* [3]. The latter is a system which there is no prior information available, and the former refers to systems where all necessary information is available. In reality, all systems are somewhere in between the two categories.

Generally speaking, it is preferable to use as much a priori information as possible. This can describe the system adequately with a set of functions. If the information is used

correctly, the model is more accurate than the black-box one. When no such information is available, functions can be used as a general means to cover all different models. Black-box models estimate both the functional form of relations between variables and the numerical parameters in those functions. One of its disadvantages is that it is difficult to estimate parameters as complexity increases. We will be using the white-box approach, and hope that we can model large scale systems more accurately and provide meaningful results.

2.4 System Biology

Understanding biological systems is a essential step to understanding diseases. The human immunodeficiency virus (HIV) [73] is a good example; HIV causes progressive deterioration of the immune system and leads almost invariably to AIDS and death from opportunistic cancers and infections. The high rate of HIV viral mutation makes developing vaccine difficult, and results in rapid positive selection for drug-resistant mutant strains. Recent results on multi-drug combinations are encouraging, but in most cases ultimately fail because of the development of drug resistance. It would therefore be useful to know how the resistance is developed before designing the drugs.

Generally speaking, the genetic and molecular data that we obtain describes a system but does not explain the behaviour. Modelling helps to provide a logical explanation, and in many instances is the only way to discover such regulatory features in the first place. It is also widely used in population problems to better understand the factors that might cause the descent or ascent. Predicting the population trends of some species, such as endangered animals or harmful predators, is helpful in making critical decisions.

Back in 1992, Keen [67] discovered the relationship between real biological systems, conceptual models, mathematical models and simulations. He stated that the result of the mathematical model is a formalization of the conceptual model in the equation form which describes the response of a system to some variables such as time or temperature. The mathematical model may also be derived from statistical analysis of the experimental

data, or by a combination of both approaches. The process of formulating the mathematical model improves the conceptual model. The feedback loop loop (See Figure 2.3) refines both the mathematical and the conceptual model.

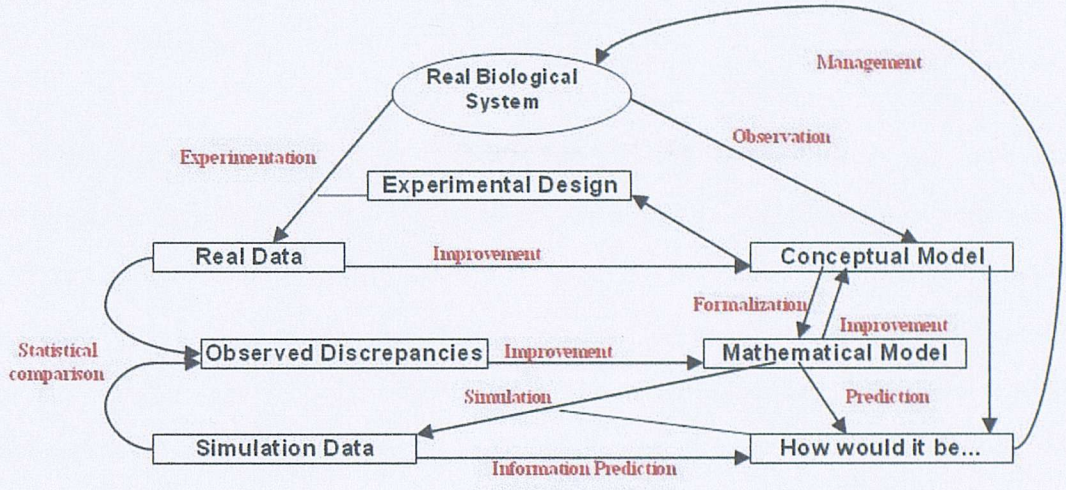


Figure 2.3: The role of modelling and simulation (taken from [67])

System Biology, an emerging field, is expected to have major impact on future biological and medical research. It aims to understand biological processes at the system level, using mathematical analysis and computational tools, and integrating the obtained information in experimental biology. It has the following characteristics: multidisciplinary research with a close interaction between experimental research (biology, chemistry, physics...); mathematical modelling and analysis (mathematics, bio-informatics, engineering..); generation and iterative improvement of mathematical models that realistically describe biological processes, and have the ability to elucidate system properties and to predict the outcome of perturbations; and use of existing and generation of novel data, in particular quantitative data, time courses and spatial information of high definition.

Hohmann etc. [59] proposed a model system which aims to develop the field, and advance our understanding of the rules and principles of the dynamic operation of

cellular systems. They plan to develop a web-based resource which allows the research community to contribute both data and mathematical models, and to simulate biological processes. In the white paper, they pointed out the importance of System Biology. The mathematical model is considered to be a tool for data interpretation and experimental planning as well as being a vehicle for exploring insights on fundamental principles behind biological systems.

Michor etc. [79] stated that evolutionary concepts such as mutation and selection can be best described when formulated as mathematical equations. In the work, they discussed how the quantitative theory of somatic mutation and selection help us to evaluate the role of genetic instability. They have reviewed models of somatic evolution with constant selection, which are based on the assumption that the fitness of a mutant neither depends on their own relative abundance nor on the relative abundance of others. Mathematical models of cancer evolution are essential for quantifying the effects of mutation, selection and tissue architecture.

In a survey of biological system simulation [69], Kim pointed out that living systems are made up of interacting chemical and physical processes, all of which can be described in mathematical terms. These processes, however, are complex and have resisted mathematical analysis by the classical methods successfully used by chemists and physicists. Biologists with simulations skills find it useful in comprehending biological models quickly, separating sense from nonsense. Unfortunately, the simulation techniques are not as familiar to practising biologists as the data analysis techniques.

Chapter 3

Modelling Indistinguishable Agents

3.1 Introduction

As mentioned before, our approach can be used to model agents at different levels of detail. According to their characteristics, we categorize the agents as either *indistinguishable* or *distinguishable*. The term *indistinguishable agents* describes those that exist in large quantities, behave identically and have trivial individual impact on the system. In the field of biology, cancerous cells are a good example of indistinguishable agents. As a result of the negligible effect that a single instance produces, only the impact of a collection of such instances to the system is noticeable. It is therefore unnecessary and impractical to model such agents separately, for we are interested in aspects such as their density.

In this chapter, we use three examples to illustrate how our approach can be applied to model open systems comprising indistinguishable agents. With the examples, we demonstrate how competing and collaborating natures are modelled and how we can make use of the obtained information.

3.2 An Example: Modelling Biological Systems

3.2.1 System Description

We first demonstrate how our approach can be used to model a system with two competing populations. The two populations are the *viruses* and the *agents*. A virus may indiscriminately attack an agent, and when it does this it either destroys the agent or is itself destroyed. On the other hand, agents can be of two types: type A (agile) agents have no immunity against viruses. If they are attacked, they have a higher chance of being destroyed than Type B (resilient) agents, which have been immunized and therefore have a smaller chance of being destroyed by a virus attack. Type B agents are immunized Type A agents. If a Type A agent attacks the virus population, it has a higher probability of destroying a virus and of escaping unscathed, while when a Type B agent attacks the viruses it has a smaller probability of successfully destroying a virus and a larger probability of being destroyed in the process. The described system can be illustrated by the figure below (See Figure 3.1).

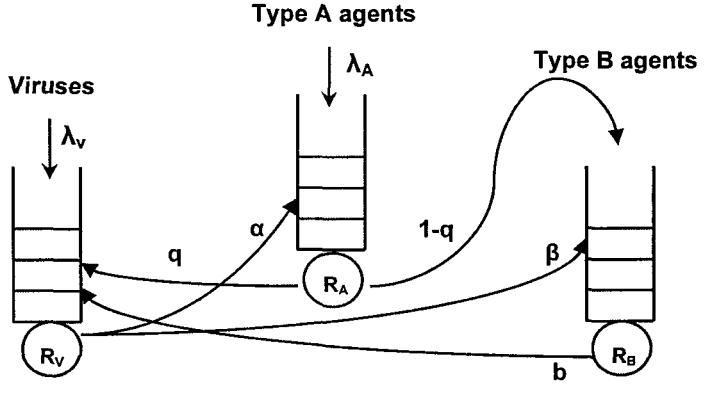


Figure 3.1: An abstract description of a system containing two competing populations

3.2.2 Model Description

The state of the system under consideration will be the vector $x(t) = (x_V(t), x_A(t), x_B(t))$, where the elements of the vector are the numbers of viruses, Type A agents and Type

B agents at time t, respectively. Note that these (discrete) numbers may also refer in certain cases to concentrations of agents (e.g. biological viruses per cubic meter, tens of computer viruses per computer node, and etc.).

Agents arrive to the system according to a Poisson process of rate λ_A, while viruses arrive in another Poisson process of rate λ_V, both of which are independent of each other. The external arrivals of agents only concern Type A agents, since a Type B agent is a transformed type A agent through the immunization process. Whenever there are viruses in the system, they will attack the population of agents at rate R_V, and will select a victim among the Type A agents with probability α, or among the Type B agents with probability β. Before deciding to attack the agents, the viruses do not verify if there are any agents in the system; a virus will be destroyed after any attempted attack. Thus, as a result of a virus attack, the attacking virus itself will be destroyed, and when attacked a Type A agent will be 'killed' with probability c, while a Type B agent will be killed with probability K.

Concurrently with this, provided Type A agents exist, they take action at rate R_A either by attacking a virus with probability q, or they choose to immunize themselves and thus mutate into a Type B agent with probability $(1-q)$. If a Type A agent attacks a virus, it will successfully destroy it with probability a, in which case either it will return unscathed to rejoin the population of Type A agents with probability e or it will itself be destroyed as well as the virus it attacked. With probability $(1-a)$ when a Type A agent attacks a virus it is unsuccessful and is itself destroyed.

Also concurrently, provided Type B agents exist, they take action at rate R_B by attacking a virus. If a Type B agent attacks a virus, it will successfully destroy it with probability b, in which case it either returns unscathed to rejoin the population of Type B agents with probability f; or with probability $(1-f)$ it is itself destroyed while destroying a virus. With probability $(1-b)$ when a Type B agent attacks a virus, it is unsuccessful and is itself destroyed.

The difference in the capability of Type A and Type B agents with respect to self-protection against virus attacks is given by the probabilities c and K, while their ability to attack successfully is given by the probabilities a and b, and their ability to remain unscathed after an attack is given by the probabilities e and f. In general, we would represent the contrast between agility and resilience of Type A and B agents as follows: $R_A > R_B$, $e > f$, and $K > c$.

3.2.3 System Equations

We assume that all the rates defined above are parameters of exponential distributions for independent random variables, so that the system we have described is reduced to an infinite and discrete state space and continuous time Markov chain. We will detail the derivation of the following equation (See Equation 3.1) in the next section. For the time being, we simply indicate that the process $x(t) : t > 0$ can be studied by writing the Chapman-Kolmogorov equations [34] for $p(k,t) = Prob[x(t) = k]$, where $k = (k_V, k_A, k_B)$ is a 3-vector of non-negative integers representing the number of viruses, Type A and Type B agents in the system, respectively.

Proposition 1 The Chapman-Kolmogorov equations for the system we have de-

scribed are:

$$\begin{aligned}
\frac{d}{dt}p(k,t) &= \lambda_V p(k-z_V,t)1[k_V>0] + \lambda_A p(k-z_A,t)1[k_A>0] + \\
&R_V(\alpha c p(k+z_V+z_A,t) + \beta K p(k+z_V+z_B,t) + \alpha p(k+z_V,t)1[k_A=0] + \\
&\beta p(k+z_V,t)1[k_B=0] + \alpha(1-c)p(k+z_V,t)1[k_A>0] + \\
&\beta(1-K)p(k+z_v,t)1[k_B>0]) + R_A(qaep(k+z_V,t) + \\
&qa(1-e)p(k+z_V+z_A,t) + q(1-a)p(k+z_A,t) + \\
&q(1-a)p(k+z_A,t)1[k_V=0] + (1-q)p(k+z_A-z_B,t)1[k_B>0]) + \\
&R_B(b(1-f)p(k+z_B,t) + bfp(k+z_V,t) + (1-b)p(k+z_B,t) + \\
&p(k+z_B,t)1[k_V=0]) - (\lambda_V+\lambda_A)p(k,t) - \\
&(R_V 1[k_V>0] + R_A 1[k_A>0] + R_B 1[k_B>0])p(k,t) \qquad (3.1)
\end{aligned}$$

where on the right hand side we show the transition rates from neighboring states into state k, and $z_V = (1,0,0)$, $z_A = (0,1,0)$, and $z_B = (0,0,1)$.

Theorem 1 When the stationary solution of these equations exists [34,36,39,40,48], it has the following form:

$$p(k) = \lim_{t\to\infty} p(k,t)(1-\rho_V)(1-\rho_A)(1-\rho_B)\rho_V^{k_V}\rho_A^{k_A}\rho_B^{k_B} \qquad (3.2)$$

where $k_V > 0, k_A > 0, k_B > 0$, and $0 < \rho_V, \rho_A, \rho_B < 1$ satisfy the following system of algebraic non-linear equations:

$$\begin{aligned}
\rho_V &= \frac{\lambda_V}{R_V + qa\rho_A R_A + b\rho_B R_B} \\
\rho_A &= \frac{\lambda_A + qae\rho_A R_A \rho_V}{R_A + \alpha c \rho_V R_V} \\
\rho_B &= \frac{(1-q)\rho_A R_A + bf\rho_B R_B \rho_V}{R_B + \beta K \rho_V R_V}
\end{aligned} \qquad (3.3)$$

Corollary 1 The total average number of agents in the system in stationary state

defined as $N = \lim_{t\to\infty} E[X_A + X_B]$ is given by:

$$N = \frac{\rho_A}{1-\rho_A} + \frac{\rho_B}{1-\rho_B} \tag{3.4}$$

3.2.4 Chapman-Kolmogorov Equations

The above Chapman-Kolmogorov Equations for the system are derived as below. We first define the probability that the system is in state k at time t to be:

$$p(k,t) = P\{N(t) = k\} \tag{3.5}$$

The probability that the system is in state k at time $t+\Delta t$ is shown below. The right hand side is composed of all the possible transitions from neighboring states into this state.

$$\begin{aligned} p(k,t+\Delta t) = {} & \lambda_V \Delta t(1-R_V\Delta t)p(k-z_V,t)1[k_V>0] + \lambda_A\Delta t(1-R_A\Delta t)p(k-z_A,t)1[k_A>0] + \\ & R_V\Delta t(1-\lambda_V\Delta t)\alpha(1-c)p(k+z_V,t)1[k_A>0] + R_V\Delta t(1-\lambda_V\Delta t)\beta K p(k+z_V+z_B,t) + \\ & R_V\Delta t(1-\lambda_V\Delta t)\alpha p(k+z_V,t)1[k_A=0] + R_V\Delta t(1-\lambda_V\Delta t)\beta p(k+z_V,t)1[k_B=0] + \\ & R_V\Delta t(1-\lambda_V\Delta t)\alpha c p(k+z_V+z_A,t) + R_V\Delta t(1-\lambda_V\Delta t)\beta(1-K)p(k+z_V,t)1[k_B>0] + \\ & R_A\Delta t(1-\lambda_A\Delta t)qaep(k+z_V,t) + R_A\Delta t(1-\lambda_A\Delta t)qa(1-e)p(k+z_V+z_A,t) + \\ & R_A\Delta t(1-\lambda_A\Delta t)q(1-a)p(k+z_A,t) + R_A\Delta t(1-\lambda_A\Delta t)q(1-a)p(k+z_A,t)1[k_V=0] + \\ & R_A\Delta t(1-\lambda_A\Delta t)(1-q)p(k+z_A-z_B,t)1[k_B>0] + R_B\Delta t b(1-f)p(k+z_B,t) + \\ & R_B\Delta t b f p(k+z_V,t) + R_B\Delta t(1-b)p(k+z_B,t) + R_B\Delta t p(k+z_B,t)1[k_V=0] + \\ & [(1-\lambda_V\Delta t)(1-R_V\Delta t)(1-\lambda_A\Delta t)(1-R_A\Delta t)(1-R_B\Delta t)]p(k,t) \end{aligned} \tag{3.6}$$

Since the derivative of $p(k,t)$ is defined as:

$$\frac{d}{dt}p(k,t) = \lim_{\Delta t\to 0}[p(k,t+\Delta t) - p(k,t)]/\Delta t \tag{3.7}$$

and using the following fact to eliminate some of the terms.

$$\lim_{\Delta t\to 0}\frac{o(\Delta t)}{\Delta t} = 0 \tag{3.8}$$

where function $o(x)$ represents asymptotically negligible, and it is defined as:

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = 0 \tag{3.9}$$

In this section, we have only illustrated how we derived the Chapman-Kolmogorov for the system that is described in Section 3.2.1. A detailed explanation of how to derive Chapman-Kolmogorov equations for systems in general can be found in [47].

3.2.5 Numerical Results

In this section we present some of the numerical results obtained from the model under different parameter settings. In all of the numerical examples, we assume that all agents leave the system at the same rate (e.g. $R_V = R_A = R_B = 1$) and there are more agents leaving than entering the system (e.g.$\lambda_V = \lambda_A = 0.99$, $R_V, R_A, R_B > \lambda_V, \lambda_A$). In doing so, we ensure that the initial conditions can lead the system to a steady state. That is to say, in the long run, the number of agents in the system converges to a non-infinite number. Through the experiments, we are exploring how different parameters affect the behavior of the model, in this case, the survival of the agents. Thus we use the total average number of agents, N, as the objective function and vary the parameter values.

We firstly examine how the probability of Type A agents attacking the viruses (q) affects the average total agent population size. Both α and β are set to be 0.5 to indicate that the viruses attack either population of agents indiscriminately. e and f are set to be 0 to indicate that both of the agents are destroyed after attacking the virus. At the mean time, we explore the effect of the viruses on the system by varying their strength (K). We observed that under the same parameter settings, the total average number of agents in the system decreases as the viruses get stronger (See Figure 3.2).

The obtained results not only reflect the behavior of the system but also can help us in deciding what to do when there are one or more options. If all the type B agents die during a virus attack ($K = 1$), then it is better for type A agents to attack the viruses rather than immunize to type B. Whereas if only a small percentage of type B agents die, then it is better for type A agents to either go through the immunization process, or to attack. Somewhere in between, the attack and immunization is not as good. The result

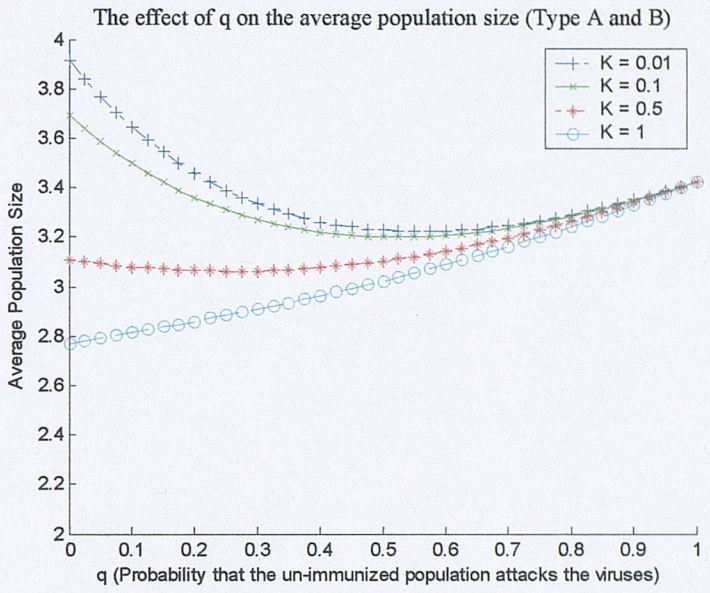


Figure 3.2: N varies as a function of the probability that Type A agents attack the viruses before being immunized

can also provide brief estimation of the parameter settings in situations where we want to keep N within a certain range. Using this system as an example, if we want to keep the number of agents under a certain level, say 3.2, all the time, then K has to be less than 0.5 and q has to be less than 0.7.

In the competing population case, a Type A agent can either be destroyed or escape after a successful attack. We modify the model to explore how N varies when half of the Type A agents escape after successful attacks (See Figure 3.3) and e is set to be 0.5 to reflect it. Comparing the two models, N increases dramatically if Type A agents can escape after attacking the viruses and the effect that K has on N remains small.

So far, the chosen parameters have reflected extreme scenarios within the model. Setting $e = 1$, $f = 0.7$, $a = 0.5$, $b = 0.3$ and piloting N against q with the same set of K, we arrive at the following results (See Figure 3.4). Differing from the previous models, the current model only has a solution for two sets of K values and within the range of 0 to 0.32 and 0 to 0.72, respectively. As mentioned in the earlier section,

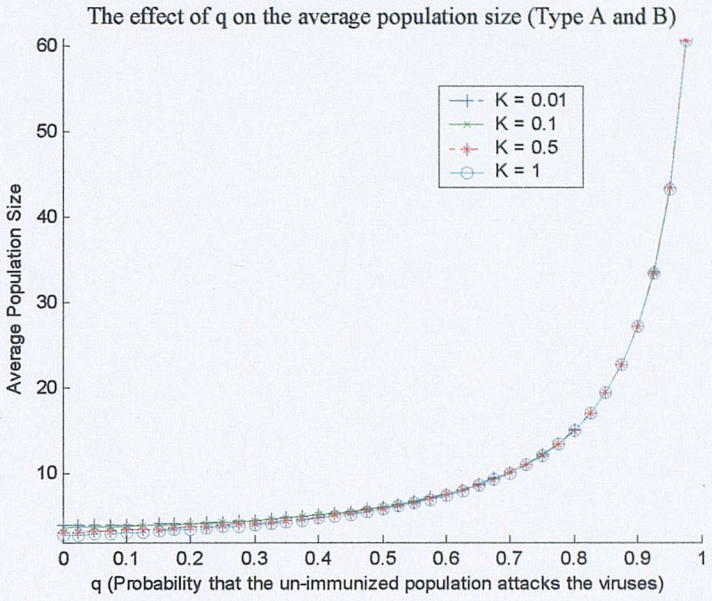


Figure 3.3: N varies as a function of q when half of Type A agents escape after attacking the viruses

the stationary solution of the non-linear equations exists when both $k_V \geq 0$, $k_A \geq 0$, $k_B \geq 0$ and $0 < \rho_V, \rho_A, \rho_B < 1$ are satisfied. We therefore examine the behavior of ρ_V , ρ_A and ρ_B when $K = 1$ and $c = 0.7$. We found that ρ_A approaches 1 as q reaches 0.72 and exceeds 1 after that. When one of the ρ_V , ρ_A or ρ_B reaches 1, the condition is not satisfied and therefore the model does not have a solution after the point $q = 0.72$ (See Figure 3.5). The same applies to ρ_A when $K = 0.5$ and $c = 0.3$.

By assigning $e = 0.5$ and $f = 0.3$, the model has a solution when $K = 0.1$ and $c = 0.05$ as illustrated below (See Figure 3.6). Comparing Figures 3.4 and 3.6, we could draw the conclusion that as e and f increase, the value of N increases drastically and reaches its asymptote at infinity faster. That N increases sharply indicates that ρ_V, ρ_A or ρ_B is reaching 1. By increasing q in much finer steps, we observed that N keeps increasing and approaches infinity; thus the peaks shown in the figures are the maximum values of N obtained by increasing q in steps of 0.01.

We then modify the model to see the effect of α (the probability that the virus

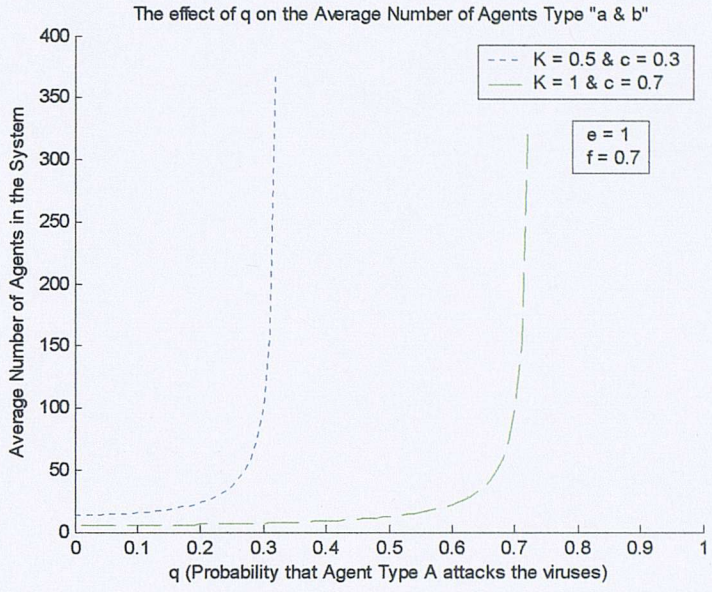


Figure 3.4: The behavior of N with different parameter settings

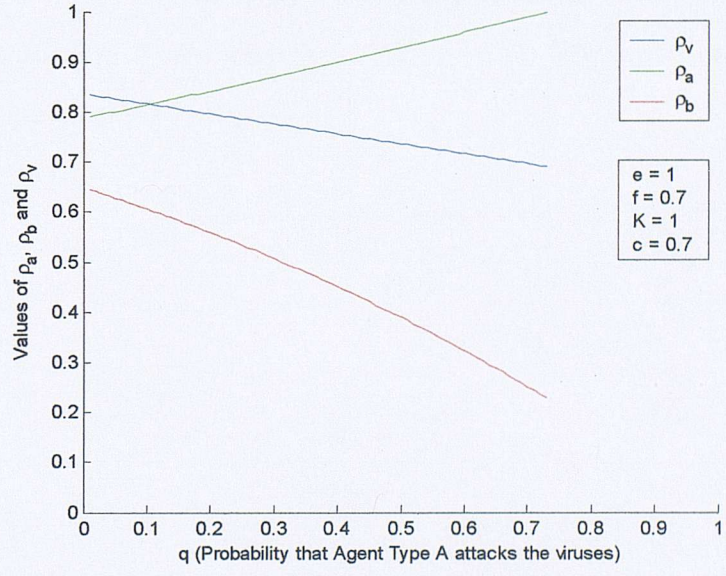


Figure 3.5: Behavior of ρ_A, ρ_B and ρ_V

attacks the Type A agents) on N. As illustrated by Figures 3.7 and 3.8, N decreases sharply as α increases. This is because the Type A agents are more vulnerable, compared to Type B agents. The effect of e and f on N is not obvious from Figure 3.7 and 3.8. In fact, N varied just above 0.3 when the f changes from 0.1 to 0.4. However, by varying q

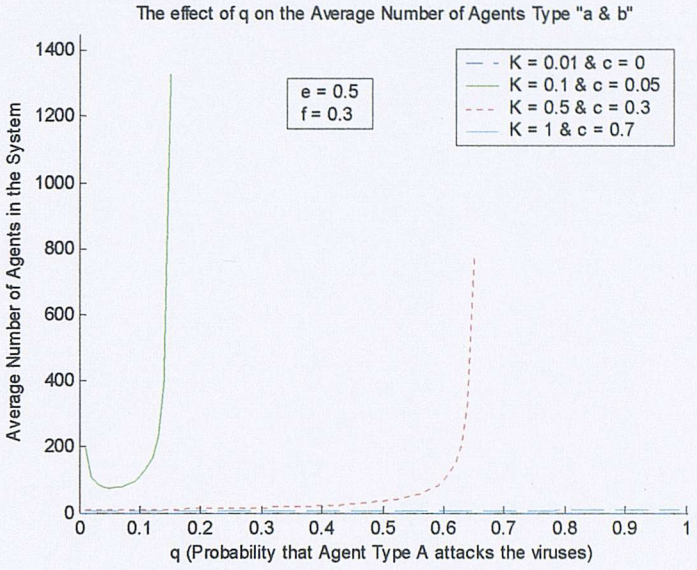


Figure 3.6: The behavior of N with different parameter settings

from 0.3 to 0.7 we see that N changes dramatically as shown in Figure 3.9. When $K = 1$ and $c = 0.7$, the value of N increased from 354 to 58153. The experiments showed that q has a stronger impact on N than α.

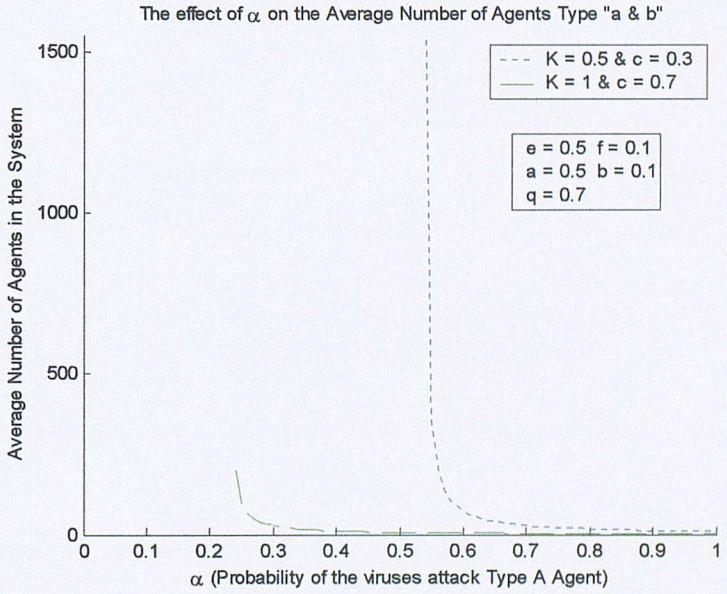


Figure 3.7: The behavior of N over a different range of α

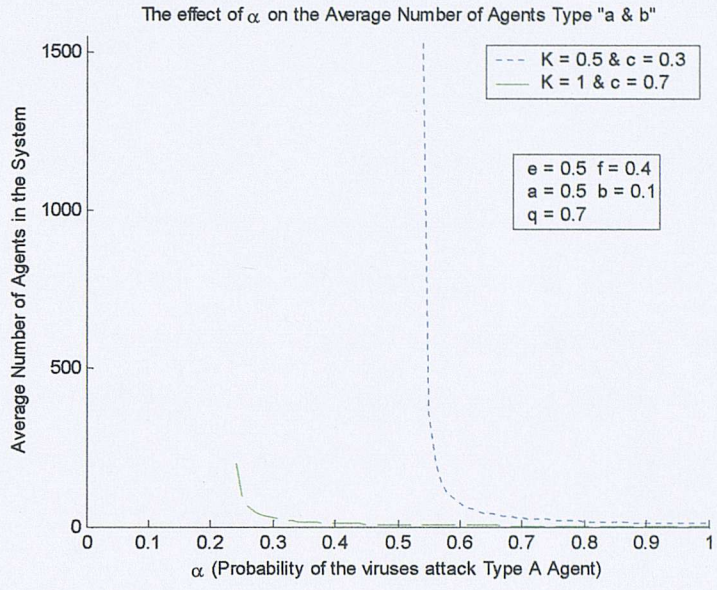


Figure 3.8: The behavior of N over a different range of α

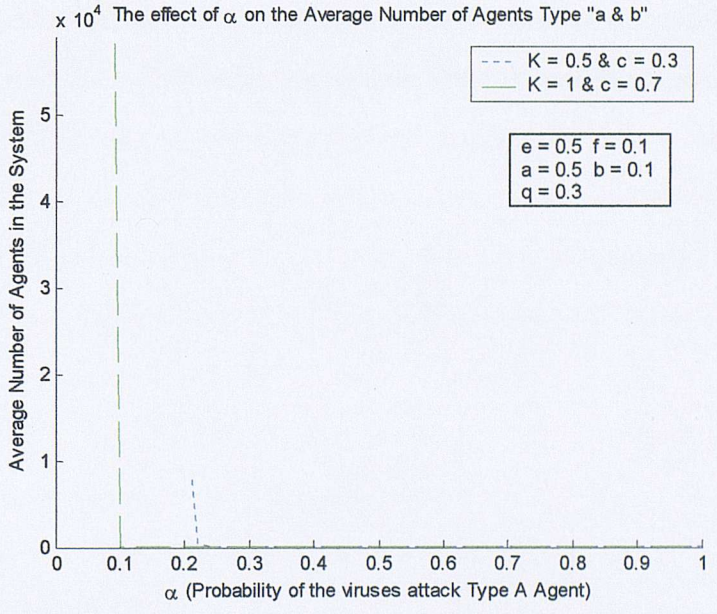


Figure 3.9: The behavior of N over a different range of α

By experimenting with different parameter settings, we are able to predict the behavior of the system, to identify the effect that the parameters have on the model and to make the right choice under the given objective function. So far, we have seen how the trade off between the agent properties affects the system, and this can be applied to

work out the optimal solutions for systems. This aspect is especially useful in situations where such trade-offs exist.

So far, we have explored the effect that different parameter settings has on the total average number of agents in the system. Results show that, by varying parameters such as probability of Type A agents attacking the viruses (q) or the probability of the viruses attack Type A agents (α), we are able to control the size of the total population (i.e.average number of agents) to a desired size. Experiments show that, for the describe scenario, there may not be a single optimal strategy (e.g. whether to increase the probability that Type A population attacks the viruses) and we often have to consider the trade-offs.

Results indicate that the total population size increases sharply when Type A and B agents have high probabilities of returning to the system after attacking the viruses (See Figure 3.2, 3.3, and 3.4). The accumulating of Type A and B agents makes it difficult for the system to settle in a steady state and we may not be able to find solutions under all parameter settings (See Figure 3.5. Comparing the effect that the probability of Type A agents attack the viruses (q) with the probability that the viruses attack the Type A agents (α), we observe that the later have a smaller impact on the system, hence the big population size.

3.3 Another Biological Example

3.3.1 Model Description

We now study another biological example where the agents may pick the wrong target and attack others by 'mistake'. The three types of cells are normal cells C, cancerous or 'bad' cells B, and chemical, radiological or immune cells A which are used to destroy the bad cells B, but which have the side effect of being able to destroy the normal cells C as well. In this model, an A-type agent will examine both B and C cells; with probability d it will correctly recognize a B-cell and destroy it, and with

probability f it will mistake a C-cell for a B-cell and also destroy it. In both cases, we assume that the A-type agent is expended in the process and that it cannot be used again. However, going beyond the requirements of the Internet related model, we also assume that C-cells can mutate into B-cells with probability π_{CB} and that the opposite is also possible with probability π_{BC}. The purpose is to model this interaction in the presence of probabilities of correct detection and false alarm on the part of agents of Type A, and to determine the dose of agent A which must be used so that most bad cells B are destroyed while the damage to cells C is limited to an acceptable level.

The state of the system under consideration will be the vector $k(t) = (k_A(t), k_B(t), k_C(t))$, where the elements of the vector are the number of autoimmune agents (expressed as an integer), of cancer cells, and of normal cells, respectively, at time t, respectively. Note that these (discrete) numbers may also refer to concentrations of the cells. The immunity defense agents into the system in a Poisson process of rate λ_A, while the cancerous and normal cells are generated in Poisson processes of rate λ_B and λ_C, respectively. All of the Poisson processes are independent of each other.

The immune cells will bind into other cells at rate R_A, and will select the type of target (normal or cancerous cell) based on the relative number of cells of each type present in the system. Thus an immune system cell will select a normal cell (Type C) with probability c, or it will attempt to bind to a cancer cell with probability b: $c = \frac{n_C}{n_B+n_C}$ and $b = \frac{n_B}{n_B+n_C}$, where n_B and n_C are the average numbers of bad B (cancerous) and normal C cells, respectively. If the cell which is targeted is cancerous (B), the immune system cell will bind to it and destroy it with probability d (i.e. it has correctly identified the nature of the cell) or it will be ineffective with probability $(1 - d)$ and the bad cell will survive. If it targets a normal cell, with probability f it will mistakenly bind to it and destroy it (the probability of a false alarm), or with probability $(1 - f)$ it will not destroy the cell. In both cases, the immune cell is expended and cannot be re-used.

Concurrently, provided cancer cells exist, they take action at rate R_B either by

killing an immune cell with probability π_{BA}, or they mutate into normal cells with probability π_{BC}. If a cancer cell attacks an immune cell, both cells will be destroyed. At the mean time, provided normal cells exist, they take action at rate R_C by transforming into cancer cells with probability π_{CB}. In a "normal" environment, we would have: $n_C \gg n_B$ (i.e. the number of cancerous cells B is much smaller than the number of normal cells C in the system), and $d > f$, i.e. the immune agent A recognizes cancerous cells more accurately than it mistakenly identifies normal cells as being cancerous. The interaction of these different types of agents is illustrated schematically in Figure 3.10.

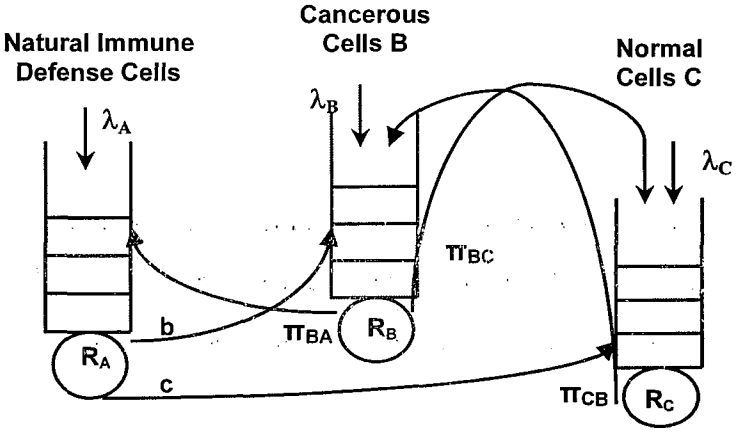


Figure 3.10: A biological model with immune system cells and cancer and normal cells

For this example, we will not repeat the Chapman-Kolmogorov equations and will only give the non-linear system equations:

$$\begin{aligned} \rho_A &= \frac{\lambda_A}{R_A + \pi_{BA}\rho_B R_B} \\ \rho_B &= \frac{\lambda_A + \pi_{CB}\rho_C R_C}{R_B + db\rho_A R_A} \\ \rho_C &= \frac{\lambda_C + \pi_{BC}\rho_B R_B}{R_C + fc\rho_A R_A} \end{aligned} \tag{3.10}$$

3.3.2 Numerical Results

To simplify the matter, we start with $R_A = R_B = R_C = 1$. We use the percentage of cancer cells in the system, N, as the objective function and vary the parameter values. Using different values of d and f, we first examine the impact of how the accuracy of the immune system in identifying the other cells affects N. Starting with a simple case, we set $\lambda_B = 0.062, \lambda_C = 0.882, \lambda_A = 0.990$. Both π_{BA} and π_{BC} are set to 0 to indicate the cancer cells neither attack the immune defense cells nor transform to normal cells. π_{CB} are set to be 0.1 to reflect that, with probability 0.1, the normal cells are mutated to cancer cells. Under these circumstances, we observe that as d and f vary, the cancer cells are within the range of 0 to 10 percent of the cell population.

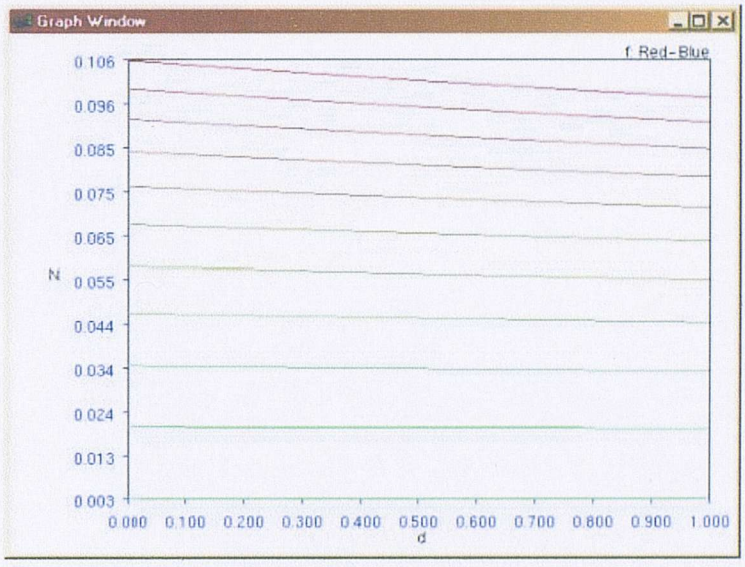


Figure 3.11: N varies as a function of d and f

In Figure 3.11, each horizontal line represents d over the range 0 to 1 in steps of 0.1. The color variation from green to red reflects the value of f changing from 0 to 1. It shows that compared to f, d has a smaller effect on the ratio of the cancer to normal cells. When the probability that immune defense cells correctly identify the cancer cells increases, N decreases, i.e. the percentage of the bad cells decreases. When the immune defense cells are falsely identifying the victim cell, N increases when more false alarms

occur.

We then set $\lambda_B = 0.007, \lambda_C = 0.882$ and $\lambda_A = 0.990$ to indicate that the cancer cells are generated at a rather low rate compared with the normal cells and the immune defense cells. π_{BA} is set to be 0.8, π_{BC} and π_{CB} are set to be 0.2 and 0.382 respectively. With this set of parameter settings, we model the situation where the normal cells have a higher probability of mutating to cancerous cells than the opposite. We examine the effect of d and f. Obviously, the higher the value of d and the smaller the value of f, the better the immune agent is performing (See Figure 3.12).

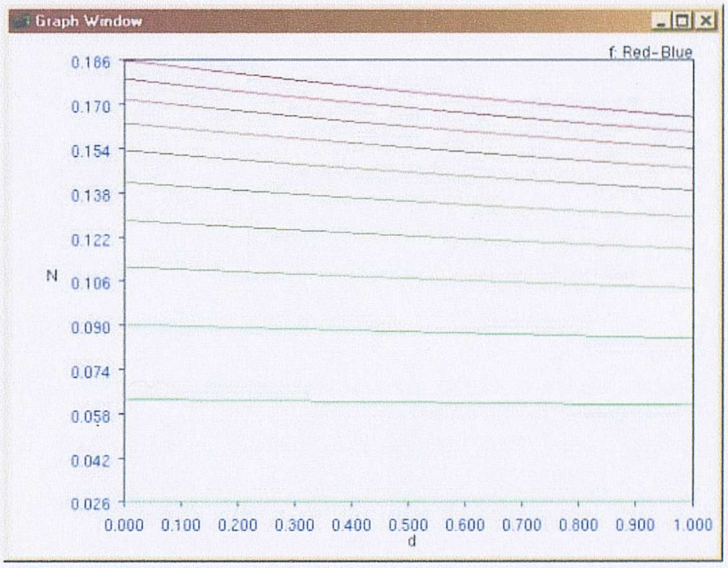


Figure 3.12: N varies as a function of d and f

Keeping all the variables the same, and increasing λ_B to 0.367, we see that the model still converges but we start obtaining invalid solutions, as illustrated by the black squares in Figure 3.13. As mentioned earlier, the stationary solution of the non-linear equations exists if $0 \leq \rho_A, \rho_B, \rho_C < 1$ are satisfied. When any one of the ρ reaches 1, the condition is not satisfied and invalid solutions are obtained.

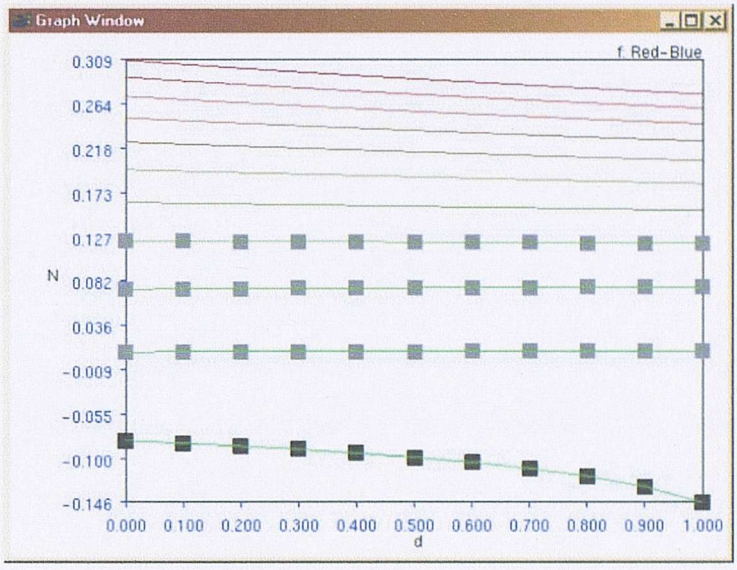


Figure 3.13: N varies as a function of d and f

3.3.3 A System Exhibiting Collaboration

The previous scenarios represent systems in which only competition is taking place; in other words agents only tend to destroy each other if they can. Of course, there are numerous practical instances where agents can both enhance and hinder each others' behavior. We will now discuss a system that exhibits not only competition but also collaboration and demonstrate how it can be solved analytically.

Consider the system where agents of type A or B are being attacked by type E agents. Type A agents, on the other hand will act upon the B cells in the following manner: any B-cell encountered by an A-cell will be 'upgraded' so that it becomes more resistant to attack by E cells. Thus B-cells will in fact belong to one of the classes B_1, B_2, so that a B_i cell is upgraded to a B_{i+1} cell after encountering an A-type cell. The upgrading will result in a probability $p_{i+1} > p_i$ that the B-type agent survives after being attacked by a E-type agent. Agents of type A and E have intrinsic rates at which they act denoted by R_A and R_E, while B-type agents have a natural death rate represented by R_B.

The characteristic parameters for each class of agents which will satisfy the following

approximate relations:

$$\rho_A = \frac{\lambda_A}{R_A + a\rho_E R_B} \ , \ \rho_E = \frac{\lambda_E}{R_E} \ , \ c_i = \frac{\rho_{B(i)}}{\rho_B} \tag{3.11}$$

$$\rho_{B(1)} = \frac{\lambda_B}{R_B + (1-a)c_1(1-p_1)\rho_E R_E + \rho_A R_A c_1} \tag{3.12}$$

$$\rho_{B(i+1)} = \frac{\rho_{B(i)}\rho_A R_A c_i}{R_B + (1-a)c_{i+1}(1-p_{i+1})\rho_E R_E + \rho_A R_A c_{i+1}} \tag{3.13}$$

where $d < 1$ allows a B-agent of class $(i+1)$ to be better protected than one of class i, while c_i is the probability that a B-agent of class i is selected by either an E-agent or an A-agent.

Let us consider a special case with $R_B = 0$. After tidying the equations, the above equations yield:

$$\rho_{B(i+1)} = \rho_{B(i)}[\delta_{i+1}] \ , \ \delta_i = \sqrt{\frac{\lambda_A}{\lambda_A + \lambda_E(1-p_i)}} \ , \ i > 1 \tag{3.14}$$

and

$$\rho_B = \frac{\lambda_B}{\lambda_A + \lambda_E(1-p_1)}[1 + \sum_{i=2}^{\infty}\prod_{j=2}^{i}\delta_j] \ , \ \rho_{B(1)} = \rho_B\sqrt{\frac{\lambda_B}{\lambda_A + \lambda_E(1-p_1)}} \tag{3.15}$$

So that, in principle, the system can be solved analytically. The probability p_i should tend to some maximum value, such as 0.8, as i increases, to represent the fact that the class of agents $B(i+1)$ is more resilient than agents of type $B(i)$, for instance $p_{i+1} = [p_i + (M - p_i).d]$ for $i > 1$, where $d < 1$ and $M < 1$.

It is interesting to consider how λ_A impacts the population of 'normal' agents B. This is schematically presented in the following figure where we see the impact of d, which modulates the resilience, as well as of the arrival rate of the 'enhancing' agents A. We have taken $p_1 = 0.1, M = 0.9$, and $\lambda_B = \lambda_E = 1$. Although the total average number of A type agents is not shown, from the curves above (See Figure 3.14) we see that they will increase very rapidly with λ_A.

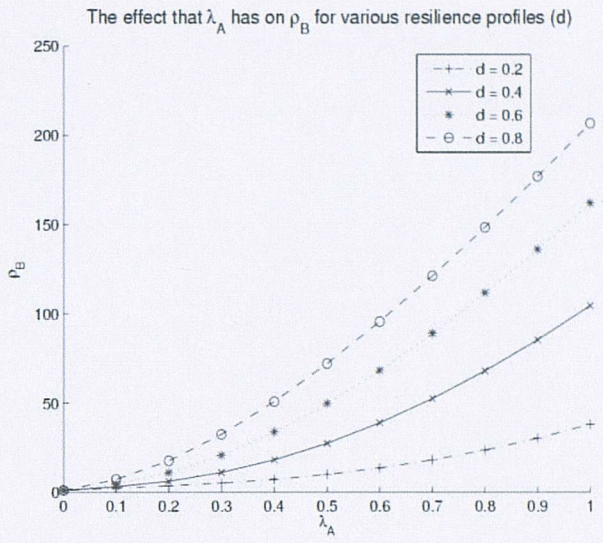


Figure 3.14: The effect that λ_A has on ρ_B for various resilience profiles (d)

There is a wide variety of practical problems related to the interaction of indistinguishable agents. Even though we have been using biological metaphors in this chapter, this does not mean that our approach is confined to this field. In computer networks for example, it is a well-known fact that undesirable or malicious network traffic of various forms, such as spam, viruses, worms, and denial of service attacks, have become very common in the Internet. As users of the Internet, we are all familiar with spam which arrives to us despite the presence of a spam blocker. Thus we can interpret the parameter d of the previous biological model as the probability that the anti-virus software correctly detects and filters the nuisance traffic and consider f as the probability that a bona fide traffic item in the network is mistakenly destroyed as a nuisance item. Similarly, we can establish mappings between the agent Types and type of traffics that we experience in computer networks and study the effect that an anti-virus software has on a computer network.

So far, we have shown that our approach is capable of modelling systems containing a large number of indistinguishable agents. The approach models behaviors such as, competition, collaboration and mutation. We also demonstrated how models can be solved numerically and analytically and they can be used to identify parameters that affect the system most. By modelling agents at a higher level of detail, the approach

provides the performance indicator at a timely manner. However, it does not give us access to an individual agent's performance. In the next chapter, we explore how systems with distinguishable agents and multiple locations can be modelled.

Chapter 4

Modelling Distinguishable Agents

In the previous chapter, we have demonstrated how to model systems that contain a large quantity of agents. Such models are highly applicable in the field of biology, computer networking and so on. In these systems, agents exist in large quantity and their individual actions have trivial impact on the system. It is therefore not necessary to pay much attention to an individual agent and its performance. Instead, we treat agents of the same nature collectively and study their performance as a whole. One of the major problems in agent simulation is the exponential increase in the computational complexity of the system with respect to the number of agents. By modelling agents collectively according to their nature rather than separately, we greatly reduce the computational complexity.

Of course, in some situations, we are more interested in the performance of a particular agent. Using urban military planning as an example, we might be interested in how a key civilian interacts with the police and the terrorist. In this case, the overall performance of all civilians in the city does not provide us with any information about the key civilian. We therefore develop models for situations where the focus of the study is the performance of individual agents or agent groups.

In this chapter, we describe the systems of interest, the components of the mathematical model as well as the model itself. Following that, we develop models for some military

urban scenarios of interest. Results show that, comparing with the simulation, the approach is a faster alternative in obtaining insights into the systems. The mathematical models are validated against the simulator [43, 44] by modelling the same scenarios and comparing the obtained results. Prior to the comparison, we discuss the differences between these two approaches. The chapter ends with a discussion of a nice property of the model that was discovered after many experiments. This property, referred to as 'modelling systems at different levels of abstraction', allows us to obtain an estimate of events happening in large scale systems without actually modelling them.

4.1 System Description

As an example of a potential application, we consider a closed system containing N distinct geographic locations and a set of C agent classes. Locations may have obstacles that an agent cannot physically enter, such as stretches of water, trees, buildings and so on. Of course, obstacles depend on the physical objects that the agents represent (e.g. land and air vehicles will have different kinds of obstacles). In this work, we do not take the local minimum problem into consideration. Hence we assume all obstacles in the terrain are convex in shape. We use the following example (See Figure 4.1) to assist us describing systems of interest. In this example, there are 2500 (50 by 50) geographical locations and 3 agent classes (team B, C and D). For the agents, trees (the scattered green dots) and buildings (the brick color rectangle) are obstacles that they can not go through.

In the systems, agents take actions to achieve their goals. A goal, be it stationary or motion-oriented, attracts agents moving towards it. A stationary goal is a specific location, whereas a motion-oriented goal exists when an agent itself is the goal (target) of others. In this case, agents in team D have a stationary goal, the 'Critical position A', which attracts the agents. On the other hand, both team B and C have motion-oriented goals. Team $C's$ task is to assist agents in team D in reaching their destination, whereas

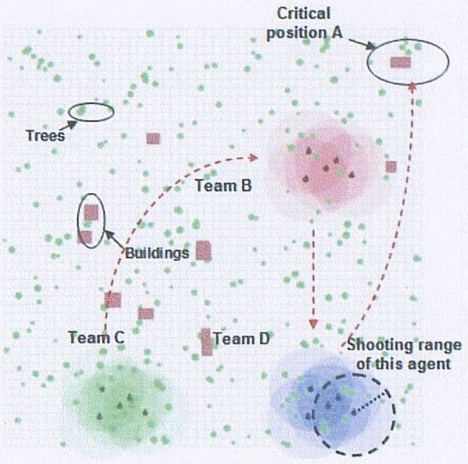


Figure 4.1: A scenario of interest

team B's goal is to destroy team D before its members complete their mission. Agents either protect or destroy their motion-oriented goals. To achieve that, agent teams might need to cooperate or compete with others. Depending on their nature, agent teams are allocated to different adversarial agent groups. Teams in the same adversarial group collaborate, whereas those in different groups exhibit competing behaviors towards each other (e.g. killing). In this case, team C and D belong to the same adversarial group, while team B is their common adversary.

In the system, an agent's motion is a straightforward representation of its intentions towards others, which are expressed in the form of forces. Forces exercised on the agents fall into one of the three categories, and they are the attractive force, the repulsive force and the long range attractive and short range repulsive force. Attractive forces are applied on an agent by other agents or its stationary goal. A repulsive force can be interpreted as the intention of moving away. For example, during their task, agents in team D avoid encountering their adversary, team B, by moving further away from it. In this case, we say that agents in team B apply repulsive forces on those in team D. On the other hand, to achieve its goal, team B has to move towards team D. Therefore, attractive forces are applied to agents in team D by team B. The long range attractive

and short range repulsive force is the key factor that groups agents of the same team together yet maintains a distance between agents. So, if two agents are too close to each other, the repulsive force is applied, otherwise, the attractive force draws them together. The strength of a force varies according to the distance between its initiator and receiver.

In the systems, agents can be completely harmless to anyone or have the ability to attack those in their adversarial groups. The main interest of this work is to study military urban scenarios; the attacking abilities are therefore interpreted in the form of shooting. Every agent is associated with a shooting range and shooting rate that specifies the effective area of its attacks and the frequency at which such attacks are launched, respectively. Obstacles are objects that agents have no access to. An obstacle can block the view of the agents and the world behind it is therefore unknown to the agents. Of course, an obstacle can be a good shelter that temporarily eliminates the danger of attacks.

Apart from moving between locations and attacking its adversaries, an agent may change its goal or nature during a task. For example, if a more urgent situation occurs and the help of team C is needed, part of it may be reassigned there to help out. We consider changing goals as a form of mutation. Also if a harmless agent is attacked too many times, it may launch counter-attacks. Such a change in nature is also considered to be mutation.

4.2 The Mathematical Model

4.2.1 Model Description

In this section, we first introduce the essential components of the mathematical model and then describe how we integrate them and form the final model. Let $i = g(t, c, k)$ denote the location of agent k of team c at time t. The location i may be a single variable or a vector, depending on the geographic representation that is being used, e.g. the x, y, z coordinates of a location on a map, where z may represent elevation

when this is relevant. Thus in a two-dimensional coordinate space, a location will be denoted by $i = (x(i), y(i))$ and a neighboring location will be denoted by some $j = i + d$ where $d \in \{(0,0), (\pm 1, 0), (0, \pm 1), (\pm 1, \pm 1)\}$. It is assumed that each agent has an initial location denoted by $S(c,k)$, and it may have (though this is not necessary) a final destination $D(c,k)$. We also assume that agents may die as a result of adversarial effects, or for other reasons, in which case they are relocated to the 'heaven' H. For this model, we assume that there is just one heaven for everyone!

The mathematical model we develop aims at being able to compute, over a large number of experiments, key quantities such as $q(i,c,k)$, the probability that agent k of team c is in location i. Such probabilities provide us with insights into the performance of agents, for example, the estimated time of an agent reaching its goal or being killed. The approach we take is based on constructing an ergodic model, i.e. one which has a stationary probability distribution, so that:

$$q(i,c,k) = \lim_{t \to +\infty} Prob[g(t,c,k) = i]. \tag{4.1}$$

4.2.2 Model Components

In the model, the motion of an agent is a result of two parameters:

- Its speed or rate of movement, $r(i,c,k)$, which will depend on the location i (reflecting the geographic characteristics of its current location), and
- The direction of the movement, which will be governed by the motion probability $p(i,j,c,k)$ that agent k of team c (will be referred to as (c,k) from now on) moves from location i to location j. In this study, we assume that locations i and j are adjacent, in which case $j = i + d$ where d is one of the eight cardinal directions from i. d also includes $(0,0)$, which means that the agent has not moved at all.

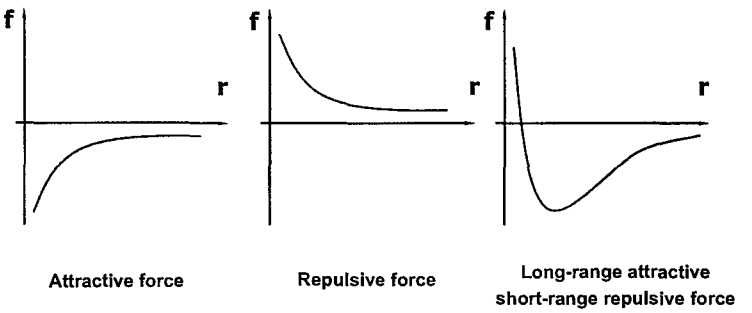
An agent's direction of motion is determined by:

- its destination, when applicable, as expressed by an attractive force, whose strength varies depending on their distance,
- the interaction between agents, as expressed by forces of attraction and repulsion,
- the physical obstacles in the terrain, or the specific terrain related difficulties which may discourage the motion in a particular direction.

The first two elements are expressed in the form of forces. So before elaborating on agents' motion, we first discuss how the forces are modelled. Depending on its initiator, a force belongs to one of the two following categories: the interagent force or the attractive force from a stationary goal. The destination of agent (c, k), $D(c, k)$, if exists, exercises an attractive force $G(i, D(c, k), c, k)$ on the agent. We use $Forces(c', k', c, k)$ to denote the interagent force that is initiated by agent (c', k') to agent (c, k). A positive coefficient implies that agent (c, k) is attracted to agent (c', k'), whereas a negative coefficient implies that agent (c, k) is repulsed by agent (c', k').

In this case, the magnitude of the forces is determined by the formula of Social Potential Fields (SPF), as shown in Figure 4.2. Attraction and repulsion are governed by the terms b/r^{β} and $-a/r^{\alpha}$ respectively. Here r is the distance between the initiator and receiver of the force. A force, with both positive a and b, is a long range attractive and short range repulsive force. Strength of the forces varies according to the linear distance between the initiators and receivers. A big α implies that the repulsive force is strong at short range and decays rapidly with distance. On the other hand, a small β implies that the attraction has long range effect.

We use an example (See Figure 4.3) to illustrate how to obtain the motion probabilities, which indicate the chances of an agent moving from one location to another, in other words, the likelihood of its motion. We first calculate the net forces, $v(i, d, c, k)$, that are exercised on agent (c, k) at direction d. The geographical location of agent (c, k), i, divides the terrain into nine sub-terrains. They are at the eight cardinal directions and the $(0, 0)$ direction (the location itself). We use the set $L(i, d)$ to represent all locations



$$f = -\frac{a}{r^{\alpha}} + \frac{b}{r^{\beta}} \qquad a, b \geq 0, \alpha > \beta > 0$$

Figure 4.2: The formula of social potential fields

at direction d from i, where d is defined as $d = direction(j - i)$. Using the sub-terrain at the north west direction as an example, we obtain the net force exercised on the agent at this direction by combining the two interagent forces and the attractive force from its goal (the star). We use the function $dist(i, j) > 0$ to represent the fact that strength of the forces changes with the linear distance between the agents. Hence, the net force exercised on agent (c, k) at direction d is:

$$v(i, d, c, k) = \sum_{all(c', k')} \sum_{j \in L(i,d)} \frac{Forces(c', k', c, k) q(j, c', k')}{dist(i, j)} + \frac{G(i, D(c, k), c, k)}{dist(i, D(c, k))} \tag{4.2}$$

Let $V(i, c, k)$ be the numerical sum of forces exerted on agent (c, k) from all directions. It can be represented as:

$$V(i, c, k) = \sum_{d \in O(i)} |v(i, d, c, k)| \tag{4.3}$$

Here $O(i)$ is the set of neighbors of i which do *not* contain obstacles. As mentioned before, agents are not aware of the situation behind the obstacles. If a neighboring location contains an obstacle, we ignore all the forces exerted on the agent at this direction. In this case, the net force at the south east direction is ignored due to the obstacle. We also introduce an *adjusting-factor* so that the model deals with situations where an agent stays at its current location. This of course raises the issue of certain agents getting "stuck" in a place from which they will not move away until conditions related to other agents have

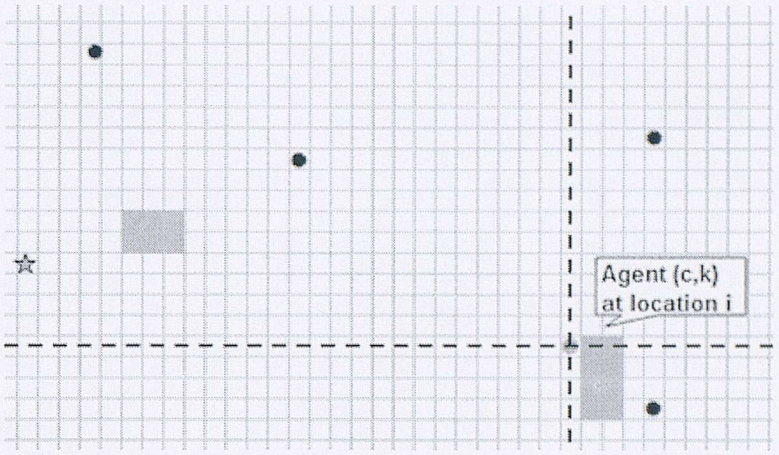


Figure 4.3: An example of how to obtain the motion probabilities

changed. The motion probability, $p(i, j, c, k)$, is defined as the following:

$$p(i,j,c,k) = \begin{cases} \frac{v(i,d,c,k)}{V(i,c,k)+adjusting-factor} & \text{if } d \in O(i) \\ 0 & \text{if } d \notin O(i), d \neq (0,0) \\ \frac{adjusting-factor}{V(i,c,k)+adjusting-factor} & \text{if } d = (0,0) \end{cases} \tag{4.4}$$

As mentioned before, apart from moving between locations, agents exhibit other behaviors such as attacking and mutating. Each agent (c, k) has a set of enemies, $E(c, k)$, that it tries to destroy, a shooting range $R(c, k)$, which determines the effective area of its attacks, and a firing rate $f(c, k)$, which is the frequency that it launches the attacks. Of course, these parameters may be identical in certain cases for all agents belonging to the same class or adversarial team. Note that in our model, the concept of enemy need not be reciprocal, i.e. $(c, k) \in E(c', k') \nRightarrow (c', k') \in E(c, k)$. Agents may also have a mutating rate associated with them. When moving from location i to j, the rate that agent (c, k) mutates to another agent (c', k') is denoted by $MutaRate(i, j, c, k, c', k')$. We assume that an agent's mutation happens during the transition of two locations.

4.2.3 Conditions under Which a Process Restarts

As mentioned earlier, the mathematical model aims at obtaining the average performance of agents in a scenario, as if it is repeated many times. In this section, we discuss the conditions that trigger the restart of the process. We consider a process (also referred to as a *game*) ends when some subset of agents, for instance any one of the agents of some class c, reaches some pre-selected set of positions, which can include their destinations. Alternatively, the game may also end when some agents reach heaven (i.e. when they are killed). To formalize the terminating conditions, we define a *final state set* $F(c,k)$ for the agent (c,k) as a subset:

$$F(c,k) \subseteq \{locations\ j\} \bigcup \{H\} \tag{4.5}$$

It is also possible that $F(c,k) = \emptyset$, which implies that this particular agent does not influence the time at which the game ends. The *terminating condition* F is now simply:

$$F = \cup_{all\ (c,k)} F(c,k) \tag{4.6}$$

and the interpretation we give to it is that:

$$At\ Time\ t\ Game\ Ends \Leftrightarrow if\ \exists\ (c,k), g(t,c,k) \in F(c,k), for\ F(c,k) \neq \emptyset \tag{4.7}$$

When a game attains its terminating condition, after some random time of average value 1 (this value is chosen for the purpose of normalization), all agents (including those that have gone to heaven), will move instantaneously to their initial locations, and the game will restart. For the purpose of this mathematical model, this cycle repeats itself indefinitely. It allows us to compute ensemble averages that are of interest. We assume that the outcomes of the scenarios are, either some agents of some designated class(es) reach their destinations, or all agents of designated class(es) reach heaven. Thus, we exclude situations where all agents become blocked and cannot move any further, or enter cycles of behavior which exclude the agents' demise, or that impair their ability to attain their destinations.

The terminating probability T is defined as the stationary probability that the model is in the terminating state:

$$T = \lim_{t \longrightarrow \infty} Prob[\vee_{all\ (c,k)} g(t,c,k) \in F(c,k)] \tag{4.8}$$

Similarly we define:

$$T(c,k) = \lim_{t \longrightarrow \infty} Prob[\vee_{all\ (c',k') \neq (c,k)} g(t,c',k') \in F(c',k')] \tag{4.9}$$

Thus $T(c,k)$ is the stationary probability that the game is in the terminating state, given that agent (c,k) is already in a final state. Suppose now that we wish to compute the expected time τ, that it takes agent (c,k) to reach some specific location γ. In this case we would set $F(c,k) = \{\gamma\}$, and the terminating probability, T, becomes $q(\gamma,c,k)$. We then have

$$\tau = \frac{1}{q(\gamma,c,k)}. \tag{4.10}$$

4.2.4 Model Equations

In this section, we describe the set of equations that represents the overall long-term behaviors of the agents. The model is built based on the equations satisfied by the stationary probability distributions of G-networks [32,36]. We heuristically, but plausibly, choose to relate the $q(i,c,k)$ to each other and to agents' parameters via the following equations:

$$q(i,c,k) = \begin{cases} \frac{Restart-Rate+Neighbors(i,c,k)}{r(i,c,k)(1-p(i,i,c,k))+Killed(i,c,k)} & \text{if } i \in S(c,k) \\ \frac{Neighbors(i,c,k)+MutationIn(i,c,k)}{r(i,c,k)(1-p(i,i,c,k))} & \text{if } i \in D(c,k) \\ \frac{\sum_{i \neq H} q(i,c,k)Killed(i,c,k)}{Restart-Rate} & \text{if } i = H \\ \frac{Neighbors(i,c,k)+MutationIn(i,c,k)}{r(i,c,k)(1-p(i,i,c,k))+Killed(i,c,k)} & \text{otherwise} \end{cases} \tag{4.11}$$

$$\begin{aligned} MutationIn(i,c,k) &= \sum_{(c',k'),d \notin O(i),d \neq (0,0)} Neighbors(i,c',k')MutaRate(i+d,i,c',k',c,k) \\ Neighbors(i,c,k) &= \sum_{d \notin O(i),d \neq (0,0)} q(i+d,c,k)r(i+d,c,k)p(i+d,i,c,k) \\ Killed(i,c,k) &= \sum_{(c,k) \in E(c',k'),j} q(j,c',k')1[|j-i| \leq R(c',k')]f(c',k') \end{aligned}$$

In addition, we use the normalizing condition which states that the sum of the probabilities that any given agent is at any locations (including "heaven") is one. Thus for each (c,k) we have:

$$\sum_i q(i,c,k) = 1. \tag{4.12}$$

You might have noticed that *Restart-Rate* is not defined in the above equation. Our approach is versatile: for an scenario of interest, it creates one set of system equations and provides insights into different aspects based on it. Hence, we only need to change the terminating condition for each aspect. For example, if we are interested in the individual impact that two particular agents reaching their goals has on the system, we simply change the termination condition without altering the model.

Now we study the set of system equations and discuss the how the model is built based on G-Networks. One distinct difference between G-Networks and the traditional queueing networks is the concept of 'negative customers'. In a traditional queueing networks, a

customer in a queue leaves the system after being served. In G-Networks, a ordinary customer can be kicked out of the system or the current queue by the arrival of a special customer, the negative customer. While calculating performance measures of the queues, for instance the utilization ρ, we have to take the elimination effects into consideration. Traditionally, ρ is represented as the ratio of the customers' arrival rate λ to departure rate r, $\rho = \lambda/r$. In the G-Networks, ρ can be obtained from the formula, $\rho = \lambda/(r + c)$, where c is the rate that the ordinary customers (also known as the positive customers) are removed from the queue. In other words, c is the arrival rate of the negative customers.

As mentioned before, we are interested in obtaining the key quantities $q(i, c, k)$, which allow us to derive agents' performance. For an agent (c, k), we first separate out special locations where the agent behave differently. They include the source $S(c, k)$, if exists, the destination $D(c, k)$ and Heaven. All the other locations in the terrain can be modelled in the same way. All the agents that are killed go to Heaven, and they will only leave there when the process is restarted.

For any location except Heaven, there are two situations under which an agent (c, k) arrives: it either enters from one of the neighboring locations or an agent of another class (c', k'), at a neighboring location, mutates to (c, k) and moves here. These two situations are represented by the term $Neighbors(i, c, k)$ and $MutationIn(i, c, k)$ respectively. When a process is restarted, all the agents will go back to their initial location. So for the source of (c, k), $S(c, k)$, there exists an additional possibility of an arrival at the rate of *Restart-Rate*. Now we take a look at the situations where agents leave a location. Apart from moving from the current location to one of the neighboring locations $r(i, c, k)(1 - p(i, i, c, k))$, an agent may also be kicked out of the system by a negative customer $Killed(i, c, k)$.

4.2.5 Dealing with Local Minima

Our approach assumes that all the obstacles are convex in shape. This is because concave shaped obstacles often cause local minima and make agents trapped at certain locations. In our model, we did not explicitly address the local minima problem. This is because,to address this problem, we can easily extended the model by implementing one of the following approaches. In [99], the authors proposed a wall-following approach which rescues the robots that fall into the local minima. Under normal circumstances, a robot operates under the potential field guide control mode. While being trapped in a local minimum, it switches to the wall-following control mode and it follows the contour of the wall. This approach does not always work at the first attempt, because a robot may fall into the same local minimum again. Another disadvantage of the approach is that, under the wall-following control mode, a robot is simply following the wall's contour and does not necessarily travel towards the direction of its goal. Of course an distinct advantage of this approach is its simplicity.

C. Q. Liu et al. [16] proposed a simple yet efficient method that recovers a robot from a local minimum by adding subgoals. Under this approach, when a robot is trapped in a local minimum, a virtual line will be draw on the contour of the obstacle so that it becomes convex in shape. Subgoals are then placed to guide the robot to move towards then along the virtual line. When the robot is around the concave obstacle, the robot starts moving towards its real goal (See Figure 4.4). Both of these approaches work based on the assumption that we have full knowledge of the terrain and the obstacles. They can be integrated into the process which motion probabilities are calculated. The motion probabilities to the locations that leads agents to local minima can be set to zero to prevent such problems occur. Another approach that does not require the terrain and obstacles information setting will be discussed in a later chapter, where we demonstrate how to model systems when we do not have access to such information.

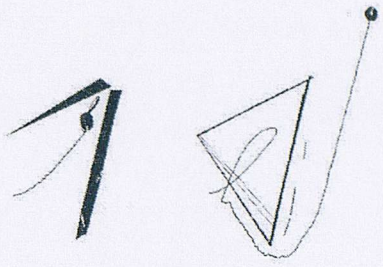


Figure 4.4: An efficient approach of avoiding local minima (taken from [16])

4.3 Scenarios and Obtained Results

To validate our approach, we start with modelling scenarios that have been studied in our group with the simulator [43,44]. By modelling the same scenarios with the simulator and the mathematical model, we compare the results and assess the correctness of the model and identify flaws if there are any.

4.3.1 Systems with a Single Agent Class

As an initial step to examine our approach, we study systems that have a single agent class. We first experiment with a 50×50 terrain with no obstacles and only one agent. The agent's starting point and destination are located at $(1, 1)$ and $(50, 50)$ respectively. The agent has to travel diagonally across the terrain to reach its goal. Since this system only contains one agent and one goal, the only force that is applied on the agent is an attractive force, initiated from its destination. With no other interference, we estimate that it will take the agent a little over 50 steps to complete the task. This is because the agent might not approach its goal in a straight line.

In this scenario, the agent does not exhibit behavior such as competing, collaborating and mutating. Hence, we remove the terms associated with such behavior, as well as the

virtual location 'Heaven'. The set of system equations is as follows:

$$q(i,c,k) = \begin{cases} \frac{Restart-Rate+Neighbors(i,c,k)}{r(i,c,k)(1-p(i,i,c,k))} & \text{if } i \in S(c,k) \\ \frac{Neighbors(i,c,k)}{1} & \text{if } i \in D(c,k) \\ \frac{Neighbors(i,c,k)}{r(i,c,k)(1-p(i,i,c,k))} & \text{otherwise} \end{cases} \tag{4.13}$$

$$Neighbors(i,c,k) = \sum_{d \notin O(i), d \neq (0,0)} q(i+d,c,k)r(i+d,c,k)p(i+d,i,c,k)$$

Since there is only one agent in this scenario, the process restarts when it reaches its goal. Hence, the departure rate of the agent at its destination is 1 and the magnitude of $Restart - Rate$ is the same as $q(D(c,k),c,k)$. The result (See Figure 4.5) shows that it takes around 53 steps for the agent to reach its goal and it is in line with our prediction. Should the scenario takes place in a bigger terrain, say 100×100, we expect the estimated time of the agent arriving its goal to be around 100 steps, as confirmed by the result (See Figure 4.6), 108 steps. Notice that after 3200 iterations in the 50×50 scenario, the model converges, whereas in the 100×100 case, it takes around 13000 iterations to converge. We will discuss the time that it takes the algorithm to converge in a later section.

Following this, we explore the effect of inter-agent forces by introducing multiple agents of the same class into the system. A 8×8 terrain is used so that the effects can be easily identified. In this scenario (See Figure 4.7), agent 1 travels from location $(2,2)$ to $(7,7)$, agent 2 goes from location $(7,7)$ to $(2,2)$, and agent 3's task is to travel from location $(5,5)$ to $(3,3)$. The inter-agent forces modelled here are long range attractive and short range repulsive.

Without any disturbance, it should take the agents around $5,5$, and 2 steps to reach their goals, respectively. We predict agent 3 to be least affected by the inter-agent forces

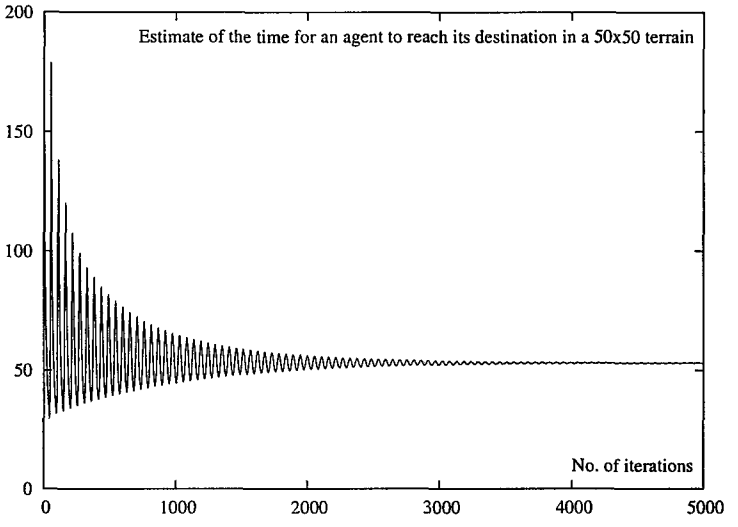


Figure 4.5: A single agent in a 50×50 terrain

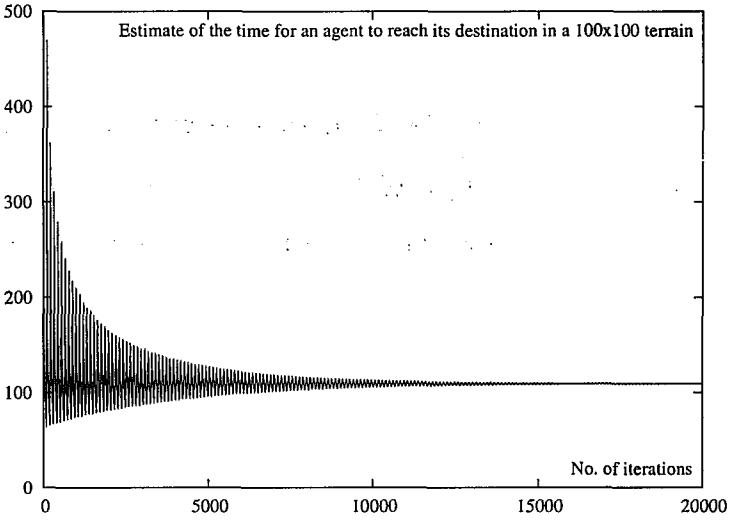


Figure 4.6: A single agent in a 100×100 terrain

since it will reach its goal before the others approaching it. Agent 1 and 2 have to pass by each other during their journey. So they attract each other before encountering, repulse when they are close-by and then attract each other again. Therefore we predict they will spend more time in the area that they encounter. The result (See Figure 4.8) suggests that the inter-agent forces have trivial effect on agent 3. Comparing with the scenario where inter-agent forces do not exist, it takes agent 1 and 2 steps longer to reach their goal.

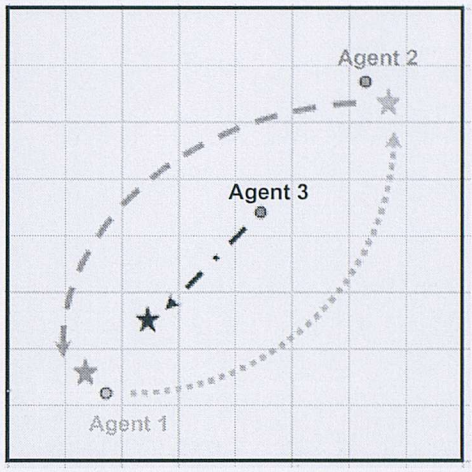


Figure 4.7: Multiple agents of the same class in a 8×8 terrain

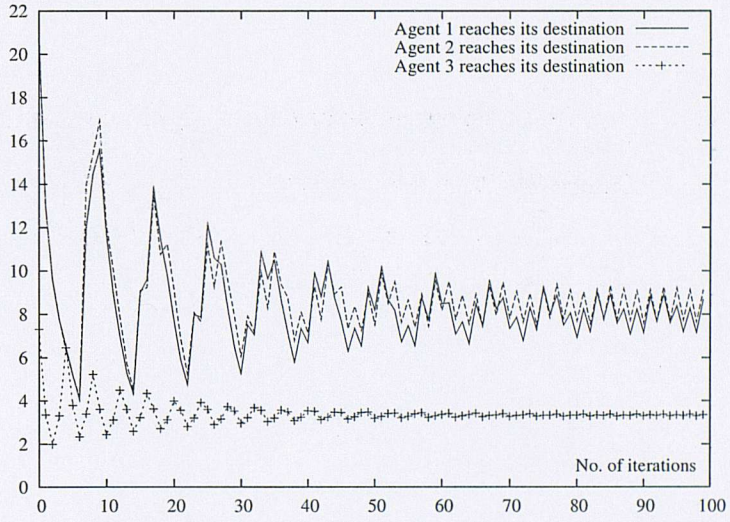


Figure 4.8: Multiple agents of the same class in a 8×8 terrain

4.3.2 Systems with Multiple Agent Classes

After the initial success, we incorporate collaborating and competing behaviors into the model by introducing multiple agent classes, in other words, the adversarial teams. We demonstrate such models with a system containing three agents of different adversarial teams, and they are referred to as the civilian (agent 1), the terrorist (agent 2) and the police (agent 3). Being completely harmless, the civilian's aim is to reach its destination

alive with the help of the police. The police attacks the terrorist to reduce the possibility of it killing the civilian. The terrorist attacks anybody who prevents it from killing the civilian. The police and the terrorist are identically equipped, in other words, they have the same shooting rate and range. This scenario demonstrates that collaborating and competing behaviors are not necessarily symmetrical.

We set the described scenario in a 15 $\times$ 15 terrain, as illustrated (See Figure 4.9). The agents' initial locations are $(2, 2)$ for the civilian, $(7, 14)$ for the terrorist and $(14, 14)$ for the police. The civilian has a stationary destination, location $(14, 14)$, whereas the police and the terrorist have motion-oriented goals. The police and the terrorist have a shooting rate of 0.2 and an effective shooting range of 3. The set of system equations for this scenario can be obtained by removing the terms $MutationIn(i, c, k)$ and $MutaRate(i, c, k)$ from the equation 4.13. In this scenario, being the focus of the study, the civilian triggers the restart of the process when it is dead or arrives at its goal. Therefore the $Restart - Rate$ takes the numerical value of $Max[q(H, c, k), q(D(c, k), c, k)]$, where (c,k) is the civilian.

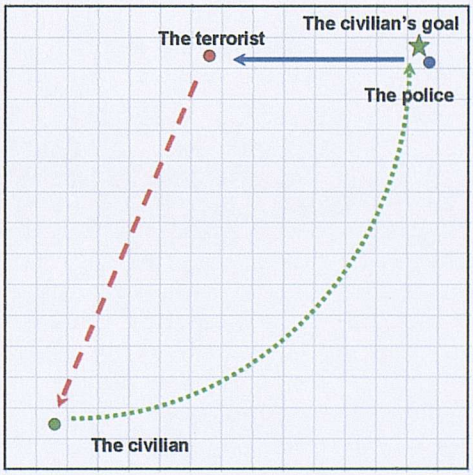


Figure 4.9: An illustration of a system of three adversarial teams

The overlapping curves (See Figure 4.10) indicate that it takes the same effort, 24

steps, for the police and the terrorist to kill each other. With the police being so far away initially, we are not surprised that the civilian is more likely to fail its task (being killed at 19 steps) than to complete it (arriving at the goal at 26 steps). For the same scenario, eliminating the police's shooting ability (set shooting rate to 0) leads to an interesting discovery: it takes the civilian 18 steps to reach its goal. This indicates that the existence of the police actually worsens the situation. In the systems, agents are not aware of the potential behaviors of the others. A terrorist only attacks a police when it is under attack or its comfortable zone (i.e. the effective shooting area) is invaded. In other words, the terrorist has a passive attitude towards the police. Hence, when the police's identity is not revealed, the terrorist launches its attacks much later. At the mean time, the civilian seizes this opportunity to go towards its goal. Of course, this does not mean that we should remove the shooting ability from the police, for results suggest that such opportunities are rare and the civilian is killed most of the time. The assistance of the police provides the civilian some chance of reaching its goal as well as the possibility of destroying the terrorist.

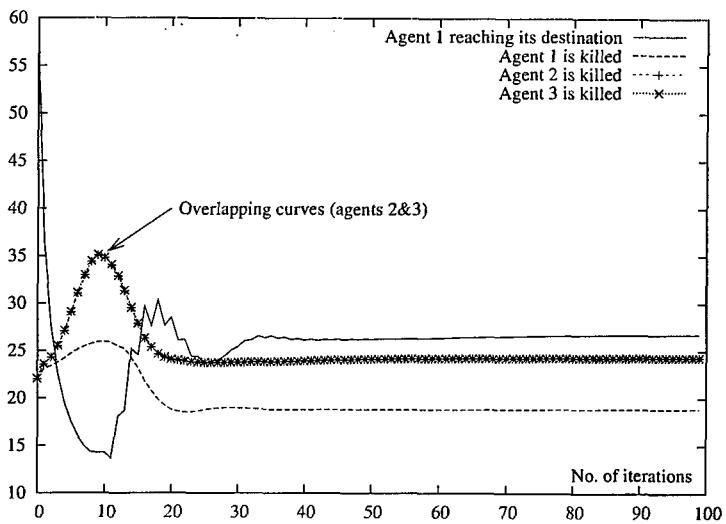


Figure 4.10: Estimated time of events happening in a system of 3 adversarial teams

So far, the police does not have a distinct advantage in helping the civilian and destroying the terrorist. We double the police's shooting rate to 0.4 and examine how that affects the outcome. Results (See Figure 4.11) show that it takes the police roughly

half of the time to kill the terrorist (15 steps) than the other way round (32 steps). Now on average it takes the civilian 22 steps to reach its goal and the terrorist 18 steps to kill the civilian. We also notice another advantage of improving the police's attacking ability: the civilian arrives its goal much faster (22 steps rather than 26 steps).

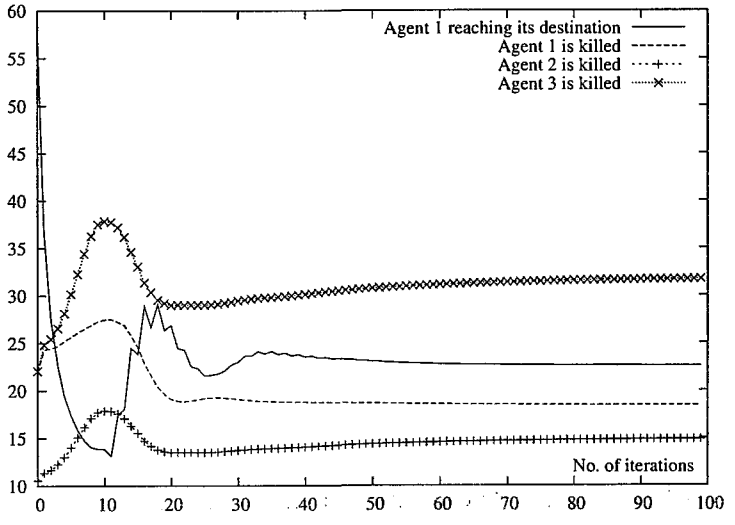


Figure 4.11: A system of 3 adversarial teams taking place in a 15 × 15 terrain

We conducted a range of experiments which vary the police's shooting rate. Results (See Figure 4.12) show that the police's shooting rate does not have much impact on the civilian's survival: the time that it takes the terrorist to kill the civilian did not change much. As the police's attacking ability increases, the civilian approaches its goal faster (from 29 steps to 18 steps). We can see that, beyond 0.4, the shooting rate has trivial impact on the time that the police kills the terrorist. It is natural that the increasing shooting rate has a positive effect on the police's self-protection.

Let the police and the terrorist have a shooting rate of 0.4, we experiment how the police's effective shooting area affects the estimated time of events happening. Results (See Figure 4.13) show that, the police is most efficient at destroying the terrorist when its shooting range is at and above 9. This setting also ensures the civilian the longest survival time in the system. With the police's shooting range changing from 8 to 9, we notice a big increase in terms of the time it takes the civilian to reach its goal. With

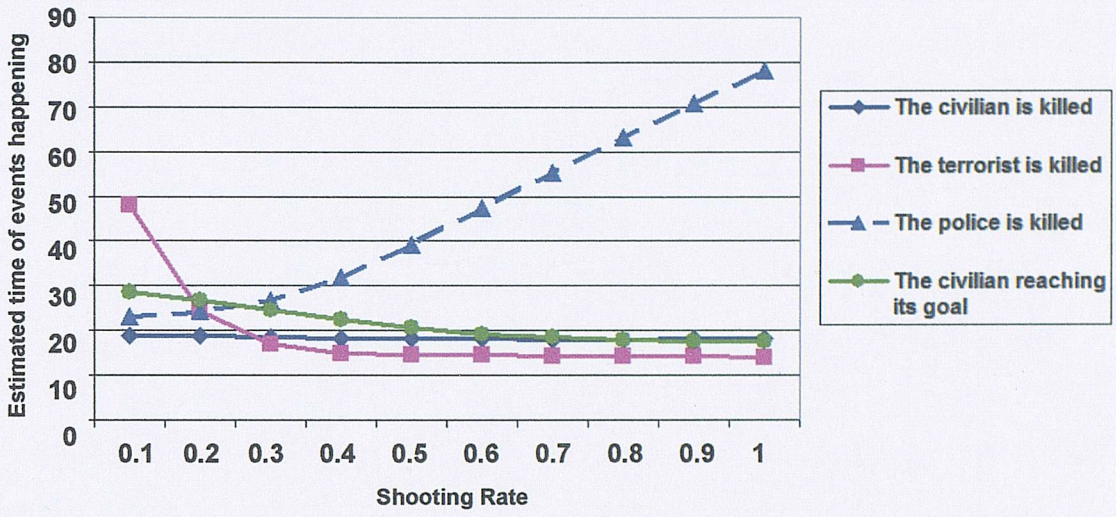


Figure 4.12: The impact of varying the police's shooting rate has on the outcome

such a shooting range, the police reveals its identity and starts attacking the terrorist at a much earlier stage. Responding to that the terrorist launches counter-attacks, however, its shooting range makes it only possible to attack the civilian for the time being, hence the increase that we have observed. Of course, this setting brings the possibilities of the police killing the terrorist before it approaches the civilian. Hence extra effort is required by the terrorist to destroy the civilian (from 9 steps to 15 steps).

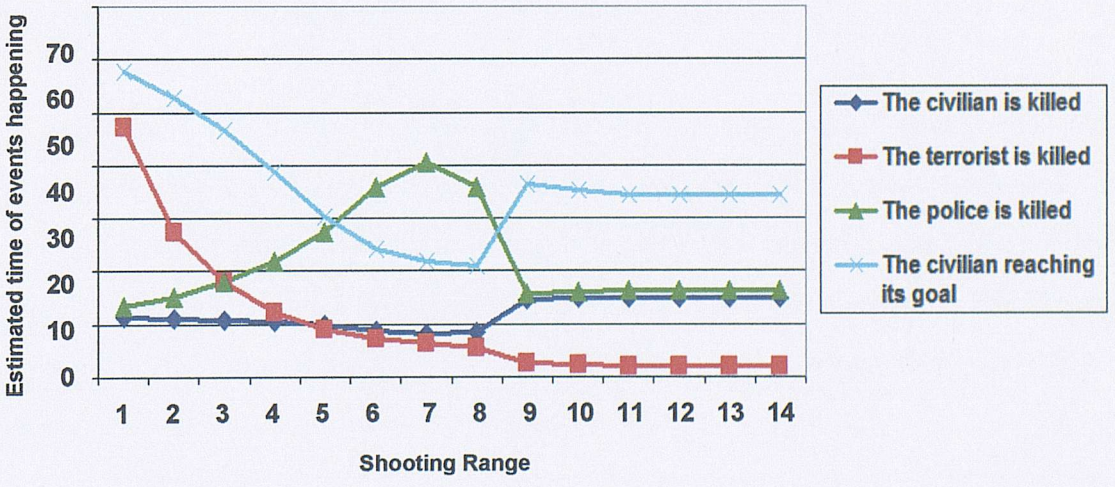


Figure 4.13: The impact of varying the police's shooting range has on the outcome

Comparing both sets of experiments, we have discovered that it takes the police similar amount of time to kill the terrorist when its the shooting range is between 1 to 4 and its shooting rate varies from 0.1 to 0.4. So far it is clear that, for the police, there is not an optimal technical setting, which offers good self-protection, delivers a fast elimination of the terrorist and allows the civilian to reach its goal quickly. Unfortunately, taking such trade-offs into consideration is a common practice in military planning. The final choice of the technical setting reflects the individual importance of the aspects.

The results indicate that the shooting rate of 0.4 and the shooting range of 8 or 9 lead to the best outcome so far. Hence we conducted experiments to see whether their combinations provide an optical solution. Results (See Figure 4.14 and 4.15) show that both combinations offer optimal solutions, where the police kills the terrorist quickly and ensure the civilian reaching its goal at a short time. With a shooting rate of 8, the police has very good self-protection, but it is only capable of killing the terrorist at 8 steps, whereas with a shooting rate of 9, the police dies much quicker, but it destroys the terrorist at around $3-4$ steps. Hence there is another trade-off that we need to taken into consideration. In both situations, the police is much better at protecting the civilian than the previous scenarios. In the field of military planning, one can use our approach to assess which parameter affects the outcome most so that the desired outcome, if exists, can be achieved by experimenting with different combinations.

Apart from the routine scenarios where the agents or agent teams collaborate and compete, the approach can also be tailored to model other scenarios where agents experience fundamental changes such as a split of force, a change of goal or even a change of nature. These phenomena are refereed to as mutation and the rate that such behaviors take place is known as the "Mutation Rate" in the model. Now we use a few examples to demonstrate how such behaviors can be modelled.

We first study the scenario where an agent team is assigned to the task of travelling from its current location $(4,7)$ to $(14,14)$. During its task, the team may be reassigned

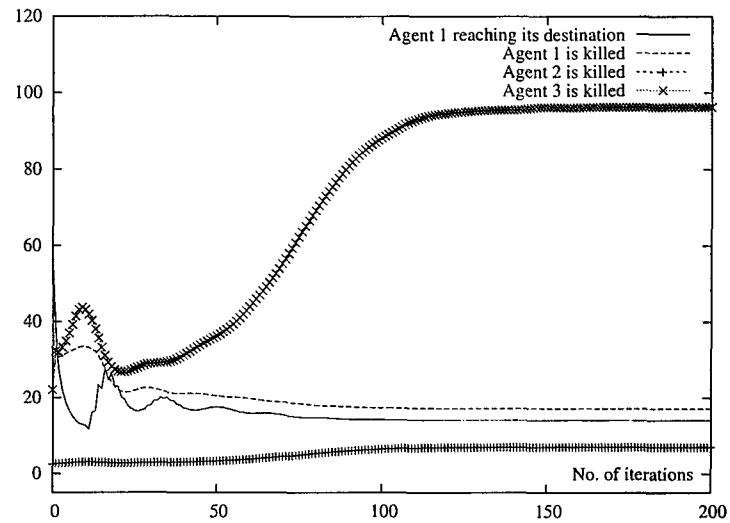


Figure 4.14: Outcome when the police's shooting rate and range are 0.4 and 8

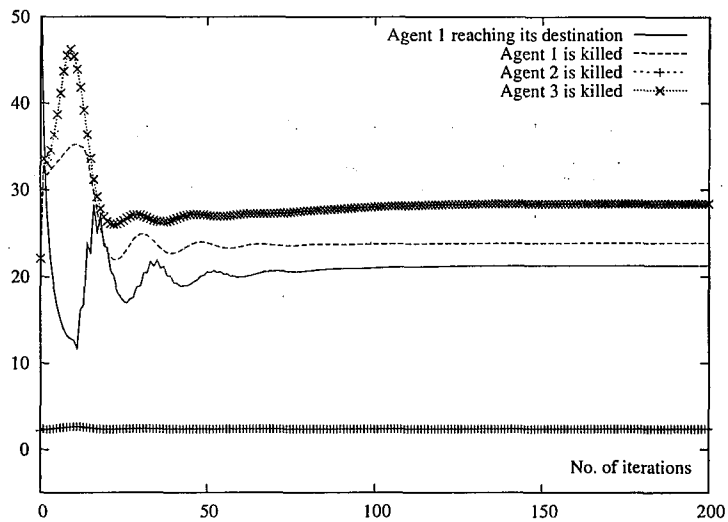


Figure 4.15: Outcome when the police's shooting rate and range are 0.4 and 9

to another task at location $(13, 2)$. We investigate how the reassigning rate of the agent team affects its performance. Such operations are often associated with an overhead cost, and we are interested in identifying a setting where the agent team accomplishes its task within the desired time frame at a low overhead. As expected, the time it takes the team to arrive at its original destination increases with the rate of it being reassigned (See Figure 4.16). We have discovered that, starting from 0.3, the reassigning rate has a trivial impact to the completion time of the new task (less than 4% of improvement in

performance with a 0.1 increase of the reassigning rate). This is because, at this point, the team has already achieved a very good performance level and it is difficult to improve on that.

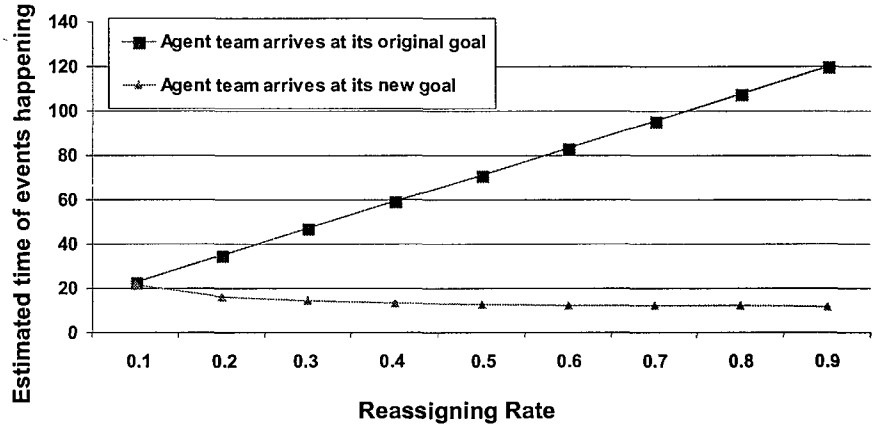


Figure 4.16: The effect of reassigning an agent team has on its task completion time

The most noticeable effect that the reassigning rate has on the agent team's performance is within the range of 0.01 to 0.2, as shown in Figure 4.17. Within this range, a 0.01 increase of the rate can bring a maximum of 40% performance improvement, whereas in the range of 0.2 to 0.9, the performance improvement varies between 0.11% to 0.01% for a 0.1 increase of the reassigning rate. At 0.09, it takes the agent team roughly the same time to complete its new or original task. This is a critical point, after which, the agent team is more likely to complete its new task than the original one. The approach allows us to identify a cost-effective plan when the number of agent teams is not sufficient for the tasks.

In the previous three agent scenarios, the civilian do not respond to the attacks that are targeted at them. We now examine the situation where the civilian gets angry and attacks back, and explore the effect of counter-attacks has on the outcome. To discover the effect that the police is equipped with a technical advantage (i.e. shooting rate) over the terrorist has on the outcome, in this scenario, we set the shooting rate of the police to be twice as big as the terrorist's. At a predefined frequency (the mutation rate), the civilian is equipped to have the same attacking ability as the terrorist.

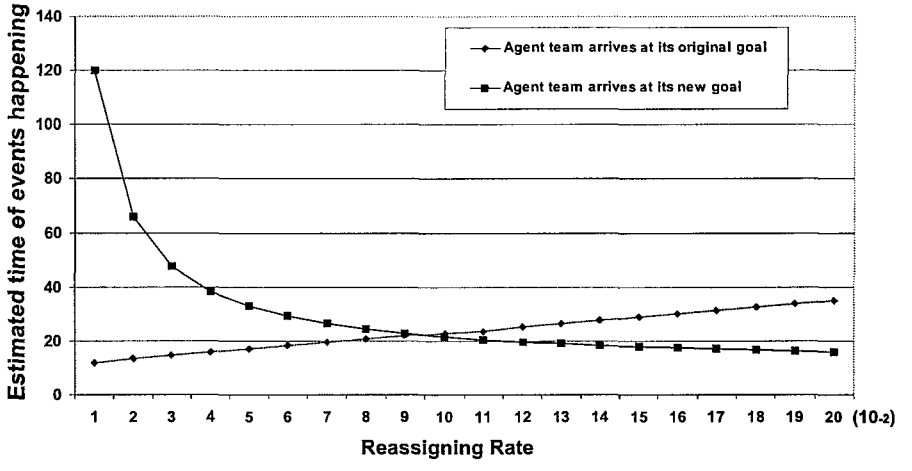


Figure 4.17: Results of the same scenario with a different range of the mutation rate

We have identified a direct link between the civilian's survival time and its attacking ability: the faster the civilian gains such ability, the longer it survives. Similar relationship applies to the police's survival time and the rate that civilian gains the ability (See Figure 4.18). The combined force of the police and the civilian keeps the terrorist's survival time to a minimum. The quicker the civilian gains the attacking power, the less the police need to interfere. Examining the time that it takes the civilian to complete its task, we notice a 20% improvement (from 26.57 to 21.09 steps) when mutation rate increases from 0 to 0.1. Comparing with the case where the civilian is incapable of launching counter-attacks, its task accomplish time has shortened. After the rate of 0.1, every increase of 0.1 of the mutation rate leads to a 6% degradation in performance. The advantage of counter-attack disappears completely when the rate is above 0.5. This is because when the civilian has none or low rate of gaining attacking power, it is danger conscious and focuses more on avoiding the terrorist and reaching its destination, whereas when it quickly obtains such power, its conflicts with the terrorist speeds up hence the negative impact on the time it accomplishes its goal. This also explains why the terrorist's survival time increases with the speed that the civilian gaining the attacking ability.

Equipping the civilian with weapons brings a considerable performance improvement.

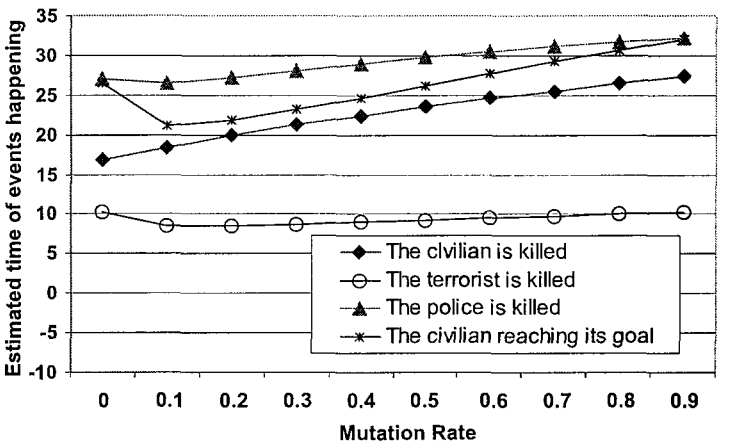


Figure 4.18: The effect that the civilian's counter-attacks has on the outcome

However, if the priority of the mission is to ensure the civilian reaching its goal before it is killed, this strategy is not as successful as adjusting the police's shooting range and rate.

We now modify the scenario and investigate how the outcome changes if the police loses its weapon during the combat. With a rate, the police's weapon becomes defective. Unaware of the situation, the police still approaches the terrorist with the aim of destroying it. Under these circumstances, the terrorist becomes the only one who possesses the ability to destroy others. We will not include the terrorist's survival time in the figure, simply because it is too predictable that the terrorist's survival time increases with the rate that the police loses its weapon.

Results (See Figure 4.19) show that when the police loses the weapon slowly, it is still able to protect the civilian and assist him reaching the goal. When the police loses its weapon at the rate of 0.1, comparing with the case where it does not loses weapon, its survival time has shortened 13.7% (22.77 rather than 26.37 steps). After this, every increase of 0.1 in mutation rate, results a 6% to 3% degradation in its survival time. When the mutation rate reaches 0.4, on average, it is more likely that the civilian reaching its goal alive than being killed. One surprising discovery is that, at this point, comparing with the case where the police has the full attacking power, its survival time has actually increased. This is because, the quicker the police loses its weapon, the easier it is for

the terrorist to attack it. While the terrorist is distracted by a weak police, the civilian takes the chance to approach its goal. A weak police, in fact, makes the terrorist wander between its two targets for longer and therefore can not manage to destroy either the civilian or the police before the civilian reaches the goal.

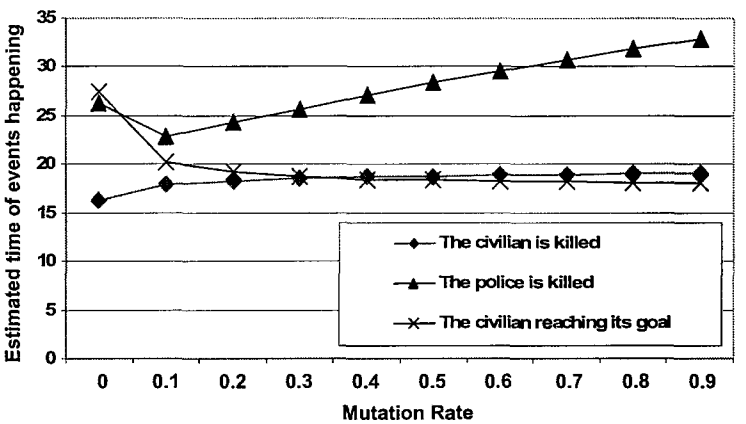


Figure 4.19: The police loses its weapon

4.4 Validating the Mathematical Model

In this work, we validate the mathematical model by studying the same scenarios in the simulator and comparing the results. Prior to the validation process, we first discuss the differences between the two approaches.

The simulator is designed for behaviorally and visually significant tactical simulations, within a physically accurate setting such as a Terrain Database. The focus of the work is goal-based navigation of a group of autonomous entities in a dangerous terrain. The design of the agent model is based on the assumption that agents will perform "outdoor" missions in a terrain containing obstacles and enemies. It is not very suitable for "indoor" missions like moving inside a building or a labyrinth, where a more specialized approach will be required. A "mission" in the agent model is defined as the problem of going from some position A to some other position B avoiding being hit by

an enemy, and avoiding the natural and artificial obstacles present in the terrain. The success of the mission is measured by the amount of time necessary for the whole group to achieve the goal and the survival rate. More detail on the simulator and its agent behavior model can be found in [66].

Both the simulator and the mathematical model are designed to provide insights into the systems, however, the means via which such information is delivered are different. The former offers visual representation of the processes as they take place, whereas the latter uses stochastic mathematical models to provide the estimated time of events happening. The agent behavior model in the simulator contains probabilistic elements, hence thousands of simulation runs is required for obtaining the average behaviors, whereas the mathematical model aims to obtain the equilibrium probabilities, so one execution is sufficient. A process in the simulator restarts at a pre-set time or time step regardless the status of the system, whereas in the mathematical model, it repeats itself when the system reaches its terminating condition.

Terrains exist both in the simulator and the mathematical model, but they are different in nature: in the former, a terrain is continuous and infinite. It takes the advantage of a visual grid, which provides the users with an idea of the progress quantitatively. In our approach, to make the computation easier, a terrain is discrete and finite. The simulator is designed based on physics, hence agents have attributes, such as mass, size, speed, acceleration rate and maximum speed. Under the net-force exercised on it, an agent accelerates to its maximum speed, travels at that pace and sometimes decelerates to avoid colliding with other agents or obstacles. Confined to its maximum speed, an agent's speed does not always reflect the inter-agent forces exerted on it. It is worth mentioning that in simulator version we used for the experiment, agents can be trapped at certain locations (local minima). Assuming an agent has reached its destination when it is within a certain range can avoid such situations happening.

The mathematical model is an abstract representation of the system, and it does not

model agents' mass and size. In order to make computation easier, we assume uniform motion. That is, the distance between two neighboring locations, for example, one of any shaded locations and $L1$ (See Figure 4.20), is 1 unit. However, distance is not always measured this way. When calculating the strength of a force, we use the actual distance between its initiator and receiver. For example, the distance between $A1$ and $A0$ is $\sqrt{2}$ whereas $A2$ and $A0$ are 1 unit length away from each other. Speed is not explicitly modelled. An agent is said to move from a location to one of its neighbors at unit time. Its 'speed' is therefore determined by the actual geographical size assigned to the location and the time unit. In the mathematical model, motion probabilities are true representations of the forces exercised on the agents, except the situations where there exists an obstacle.

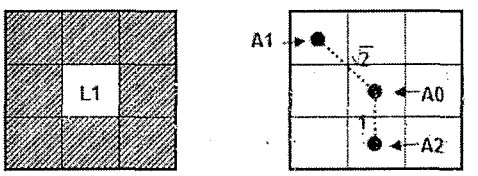


Figure 4.20: Different ways of measuring distance

Attacking is modelled differently in the two approaches. In the simulator, an agent fires with a pre-defined rate at a randomly chosen enemy in its effective shooting area. The outcome of shooting is determined by a probability, which reflects the ratio of the distance between the agent and its enemy and its shooting range. In our model, we assume that the shooting rate of an agent reflects its successful shootings. An agent picks its nearest enemy in its effective shooting area as the victim. Also, the effective shooting area is calculated differently. Using the shooting range r as an example, in the simulator, the effective shooting area is πr^2, whereas in our model, the area is $4r^2$.

We use table 4.1 to conclude the differences of the two approaches that are discussed above. After discussing the differences between the two approaches, we simulate the scenarios that have been studied with the mathematical model. By comparing the results,

we aim to validate the mathematical model and discover potential problems if there are any. We use the obtained results (See Table 4.2) of the three adversarial teams scenario (See Figure 4.9) as an example to discuss their similarities and discrepancies. All of the simulation results are obtained from 5000 simulation runs. As you might have noticed, the magnitude of the results are rather different. The result in the simulator is presented in time steps whereas the estimated time obtained from the mathematical model is in time unit, hence the difference in magnitude.

Table 4.1: The differences between the simulator and the mathematical model

Aspects	The Simulator	The Mathematical Model
Animation	Yes	No
Obtain statistics	Many runs	1 run
Terrain	Continous and infinite	Descrete and finite
Grid	Visual	Has actual meaning
Agent attributes	Consider mass, size and speed	Not considered
Agent speed	Varies depending on situation	Uniform motion
Distance	Absolute distance	Varies (See Figure 4.20)
Attacking outcome	Probabilistic	Pre-defined under attacking rate
Attacking target	Randomly chosen within range	Nearest enemy within range

Table 4.2: Obtained results for the 3 adversarial teams scenario

Scenarios Vs. Approaches		Estimated Time of Events Happening			
		The terrorist is killed	The police is killed	The civilian is killed	The civilian reaches its goal
SRP=0.2	Simulation	445	478	299	668
SRT=0.2	Mathe.Model	24.30	24.30	18.86	26.71
SRP=0.4	Simulation	407	439	275	660
SRT=0.2	Mathe.Model	14.95	31.90	18.36	22.48
*SRP: the shooting rate of the police *SRT: the shooting rate of the terrorist					

Now, with the simulator, we study the scenarios that have been modelled with the mathematical model in the previous section, and compare the obtained results. The following table lists the scenarios that are modelled (See Table 4.3).

In the first scenario, the police and the terrorist are identically equipped: they have a shooting rate of 0.2 and a shooting range of 3. The simulation results suggest

Table 4.3: List of scenarios that are studied in the simulator

$P_{in} : (14, 14), T_{in} : (7, 14), C_in : (2, 2)$ $C_{old} : (14, 14), SRP = 0.2, SRT = 0.2, SRANGP = 3, SRANGT = 3$
$P_{in} : (14, 14), T_{in} : (7, 14), C_{in} : (2, 2)$ $C_{old} : (14, 14), SRP = 0.4, SRT = 0.2, SRANGP = 3, SRANGT = 3$
$C_{in} : (4, 7), C_{old} : (14, 14), C_{new} : (13, 2), C_{mr} : (0.1 - 0.9)$
$P_{in} : (14, 14), T_{in} : (7, 14), C_{in} : (2, 2) C_{old} : (14, 14), C_{mr} : (0 - 1)$ $SRP = 0.2, SRT = 0.2, SRANGP = 3, SRANGT = 3, SRCi = 3, SRANGCi = 3$
$P_{in} : (14, 14), T_{in} : (7, 14), C_{in} : (2, 2), C_{old} : (14, 14)$ $SRP = 0.2, SRT = 0.2, SRANGP = 3, SRANGT = 3, P_{mr} : (0 - 1)$
$*P_{in} : the initial location of the police$ $*T_{in} : the initial location of the terrorist$ $*C_{in} : the initial location of the civilian$ $*C_{old} : the original destination of the civilian$ $*C_{new} : the new destination of the civilian$ $*C_{mr} : the rate that the civilian is reassigned a task/change nature$ $*P_{mr} : the rate that the police is losing its attacking ability$ $*SRP : the shooting rate of the police$ $*SRT : the shooting rate of the terrorist$ $*SRCi : the shooting rate of the civilian$ $*SRANGCi : the shooting range of the civilian$ $*SRANGP : the shooting range of the police$ $*SRANGT : the shooting range of the terrorist$

that the police have a small advantage in this case: it takes the police (445 time steps) and the terrorist (478 time steps) to destroy their rival. The small difference is a result of the fact that, in the simulator, an agent's attack only affects the victim that it has picked. In other words, the attacks that the terrorist launched towards the civilian do not affect the police. On the other hand, all the valid attacks that the police launched have impact on the terrorist. Hence, despite being equally equipped, the police has a small advantage over the terrorist. In the mathematical model, an agent's attack affects everyone around it. Hence the police and the terrorist have equal advantage.

The simulation results suggest that when we increase the police's shooting rate, from 0.2 to 0.4, the police dies at 439 time steps, rather than 478 time steps. This is not surprising because the earlier the police attacks the terrorist, the earlier the terrorist launches counter attacks. Under this setting, the terrorist is killed at 407 steps, rather than 445. The civilian's performance experiences a trivial improvement after the

police force doubled its attacking ability. So far, it seems that the improvement did not have a noticeable impact on the outcome. However, after studying the number of times that the events happen, e.g. the number of times that the civilian dies, we realize that, for the simulator results, the average time steps of events happening is insufficient.

Before the improvement, out of 5000 simulation runs, the police (dies 2301 times) has a small advantage over the terrorist (dies 2699 times). The increasing shooting rate ensures the police successfully destroys the terrorist more than 70% of the time (3529 times). It suggests that the police is twice as likely as to kill the terrorist than the other way around. Hence it is similar to the result obtained in the mathematical model. Despite the improved protection, the civilian's performance did not improve much, the average time of arrival at its destination rises from 660 to 668 time steps. However, it arrives 170 times more than before (2886 times rather than 2716 times). This conclusion coheres with the one that we drew from the mathematical model.

We have also conducted simulations for the scenario that, during its task, the agent team may be reassigned to another task at a different location (See Figure 4.16). Similar to the mathematical model, the simulation results (See Figure 4.21) suggest that, starting from 0.3, the reassigning rate has a trivial impact on the completion time of the new task (less than 3% of improvement in performance with a 0.1 increase of the reassigning rate). At this point, the agent team only arrived at its original twice out of 5000 simulation runs (See Figure 4.22). The measured time of the agent team arrives at its new goal is in line with the estimated time of such events happening that is obtained from the mathematical model. They both suggest that, when the reassigning rate reaches 0.3, the agent team reaches a good performance level, and it is difficult to improve on that.

We then simulated the scenario where the civilian gains the attacking ability during the combat (See Figure 4.18). We have observed that the civilian's new technical advantage does not have a strong impact on its performance, and most important of all, the newly-gained attacking ability did not improve on its performance (See Figure 4.23).

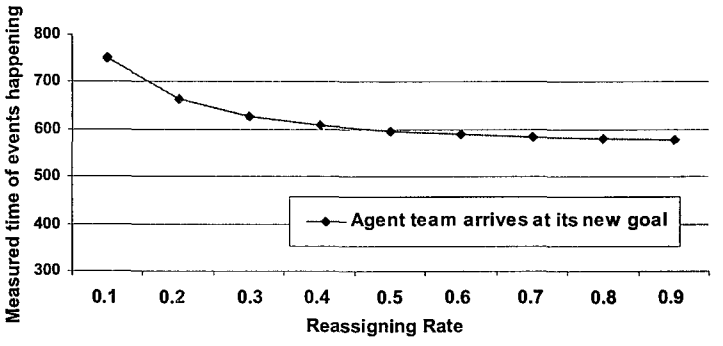


Figure 4.21: The civilian changes its goal (simulation results)

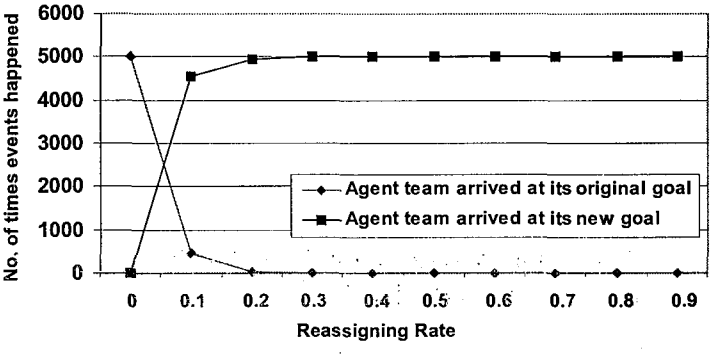


Figure 4.22: The civilian changes its goal (number of times events happened)

When the mutation rate reaches 0.2, the civilian's survival time has shortened 7% and its task accomplish time has increased by 10%. Of course, such technical advantage does not really set back the civilian's performance. Examining the results (See Figure 4.24) again, we notice that the number of incidents in which civilian is killed goes down by 24% (from 2238 times to 1710 times). Out of 5000 simulation runs, the civilian experiences a 20% increase in the number of times it reaches the goal. The fact that the civilian gaining the attacking power speeds up the encountering process of the civilian and the terrorist. This is why the civilian dies sooner than the case where it has no attacking ability.

When the mutation rate is 1, the combined force of the police and the civilian reaches its peak performance: comparing with the case where the civilian can not attack at all, the terrorist's survival time has decreased 13.4% (from 485.5 time steps

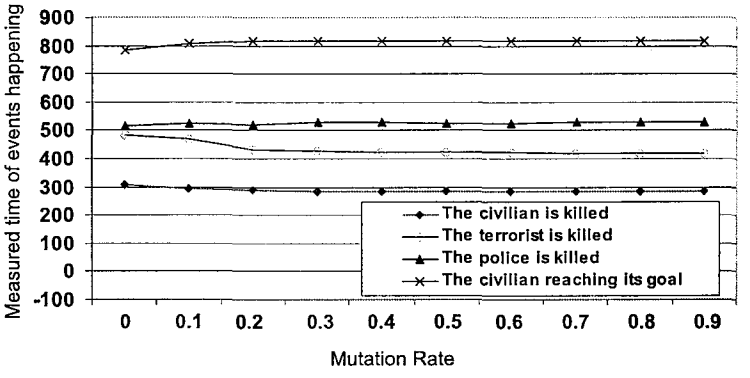


Figure 4.23: The civilian gains attacking ability (simulation results)

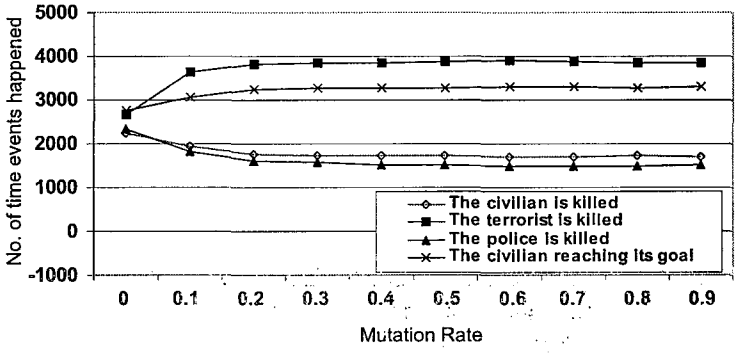


Figure 4.24: The civilian gains attacking ability (number of times events happened)

to 420.17 time steps). The combined force increases the number of incidents in which terrorist is killed from 2681 to 3853 times (out of 5000 simulations). With a maximum improvement of 2.4%, the current situation has little impact on the police's survival time. However, the combined force has a positive impact on the number of incidents that the police survives (from 2681 times to 3498 times). For this scenario, even though the results obtained from the mathematical model and the simulation are not as coherent as the previous case, both approaches suggest that a mutation rate of 0.2 is a good choice. At this rate, the system has reached a good performance level and further increases in the mutation rate can hardly lead to any noticeable performance improvement.

Now we discuss the simulation results of the scenario where the police loses its weapon. Results (See Figure 4.25) show that this change has little impact on the civilian's performance. Surprisingly, when the police loses its attacking power, its survival time

has increased. In the simulator, an agent's decision of attacking is probabilistic, and on average, an agent launches its attacks at a pre-defined attacking rate. However, if an agent is under fire, it launches counter-attacks actively. Losing its attacking power, the police attacks the terrorist at a much later time, if at all. Therefore, the counter-attacks that are targeted at the police are delayed. Of course, from the number of incidents that the police has been killed, we observe that increasing the mutation rate after 0.3 can not lead to a noticeable impact. This is because, at this point, the police is killed 4972 times out of 5000 simulation runs (See Figure 4.26).

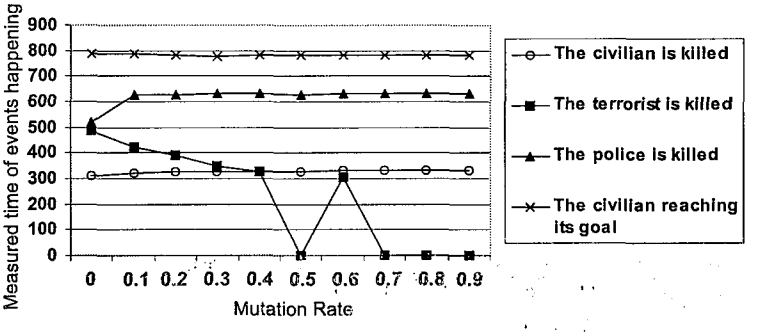


Figure 4.25: The police lost its weapon (simulation results)

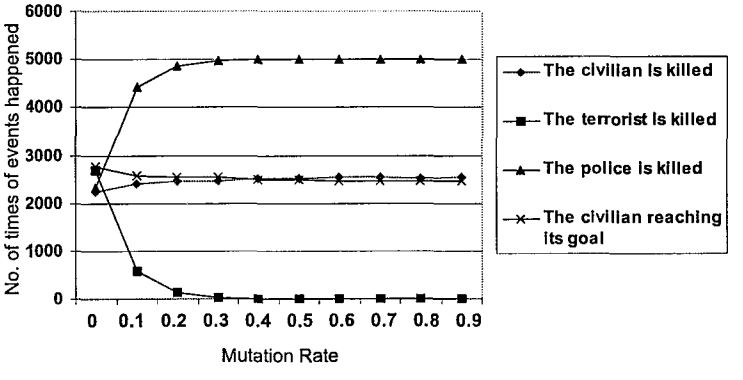


Figure 4.26: The police lost its weapon (number of times events happened)

With the range of 0.4 to 0.7, we notice some fluctuations in terms of the time that the civilian is killed. The measured time drops from 325.5 time steps to 0, then goes back to 307 and settles at 0 time steps. This is because, at the rate of 0.4, out of 5000 simulation runs, the terrorist was killed 4 times. At 0.5, it managed to survive all the

time, and at the rate of 0.6, it was killed once. The results from the simulator are the average measured time of events happening. Even though such event only happened a few times, its average is calculated. When the value of the mutation rate is above 0.7, the terrorist survived all the simulation runs, hence, the measured time of it being killed is not calculated (0 in this case). In this case, both the simulation and the mathematical model suggests that, after the mutation rate of 0.4, the civilian is more likely to reach its goal than being killed.

For the last two scenarios that we have simulated, we have noticed some discrepancies between the results obtained from the simulator and the mathematical model. This is partially caused by the fact that, to gain the whole picture of the scenario in the simulator, we have to look at both the measured time of events happening and the number of times that events happened. It is difficult to combine the results of these two aspects and compare with the one obtained from the mathematical model. In addition, the fact that shooting is modelled differently is also a factor that contributes to the discrepancy. Despite the discrepancy that we notice, both approaches discover the same or similar cost-effective settings that allow the agents reaching a good performance level.

4.5 Modelling Systems at Different Levels of Abstraction

So far, we have demonstrated an approach, that models large scale agent systems containing multiple geographical locations and provides estimated outcomes at low computational cost. After analyzing the results of many experiments, we have discovered that, the estimated outcomes of the same scenario that is set in different sized terrains are roughly proportional. That is to say, to obtain the estimated outcome of a scenario that is set in a large scale system, we do not necessarily have to model it. Instead, we can model the same scenario that is set in a much smaller terrain, and calculate the estimated outcome according to the difference in terrain size. Of course, these terrains have similar characteristics, for example, the density of the obstacles.

In this section, we will illustrate this property with results of a few scenarios that are set in different terrain sizes and explain why the approach has such a desirable property. We will not discuss the scenarios where the agents or agent teams only take actions to reach another location. Without any interference, it is not surprising that the estimated time of reaching the goal is proportional to the size of the terrain that the actions are taken place. By exploring the more interesting cases where the agents interact with each other, we investigate whether we can make use of this property and cut down the time of obtaining estimated outcomes for large scale systems.

We have chosen a few scenarios and modelled them in terrains of different sizes. In the results (See Figure 4.27, 4.28, 4.29, 4.30), the points are the obtained estimated outcome and the dash lines are the linear trend lines that are added to assist us identifying the relations. We have noticed that the estimated outcome of events of happening increasing roughly linearly as the size of the terrain increases.

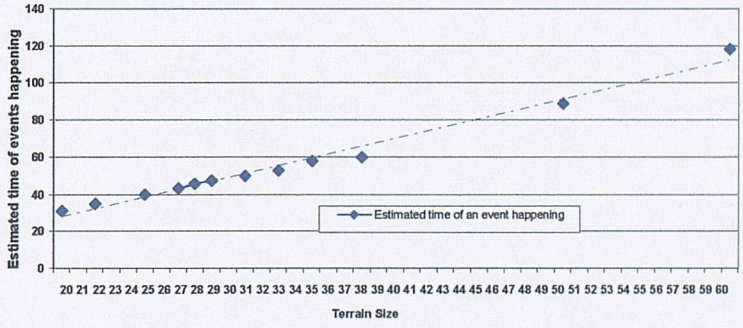


Figure 4.27: Example 1: Outcomes are roughly proportional to the terrain size

In the mathematical model, the motion probabilities of an agent are determined by the ratio of the numerical value of the net-force exerted on it from a certain direction to the numerical sum of forces from all directions. Equation 4.2 suggests that the strength of inter-agent forces varies according to the distance between their initiators and the receivers. However, the ratio remains constant. That is to say, for the same scenario, an

Figure 4.28: Example 2: Outcomes are roughly proportional to the terrain size

Figure 4.29: Example 3: Outcomes are roughly proportional to the terrain size

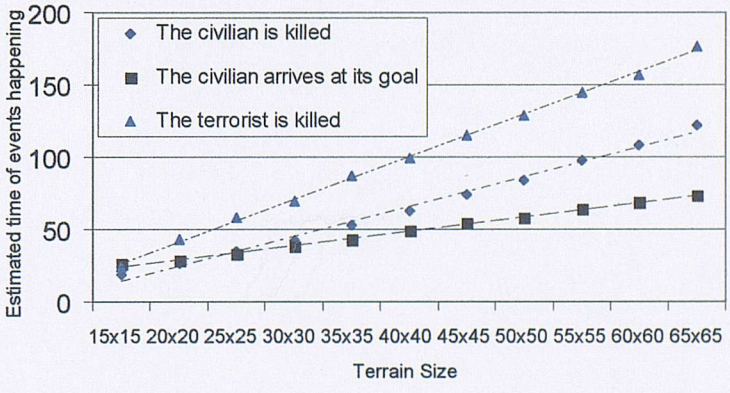


Figure 4.30: Example 4: Outcomes are roughly proportional to the terrain size

agent's motion probabilities do not change with the size of the terrain that the scenario is taking place. At the moment, we only allow the agents to move to its adjacent neighbors. Allowing agents to move to locations that are a few units away from their current

locations can be treated as the agents picking up speed. It may also mean that the area of the locations has been expanded and a new location is an integration of a few old locations.

Of course, depending on the requirement, different users may want to scale down the terrain differently. Hence, there is no fixed rule regarding how the desirable property introduced in this section should be used. Using this property requires the users to take the following two steps: scale down the terrain; calculate the agents' new initial locations and destinations based on the scale down ratio.

Generally speaking, when scaling down the terrain that the scenario is taken place, one should keep the ratio of the terrain's width and height. For example, if the size of the original terrain is a 100×80, it is better to keep the 5 to 4 ratio. Scaling down the terrain to other ratios implies that the agents' horizontal and vertical speed are different, which causes inconsistency. It is true that the smaller the new terrain is, the shorter it takes to model the scenario. However, for the results to make sense, users should make sure that agents have enough time to interact with each other in the new terrain. An extreme example would be scaling a 100×100 terrain to a 1×1 terrain. Even though the computational complexity is greatly reduced, the agents will have no time to interact with each other and the obtain results therefore makes no sense.

When scaling down the system, we often lose precision in terms of the agents' location. For example, when scaling down a 64×64 terrain to a 8×8 terrain, locations $(33, 60)$ and $(39, 60)$ will both be mapped to $(5, 8)$. This of course affects the model's precision, but in return, we are able to obtain the estimated outcomes much quicker. Once the mapping is done, we can simply model the new scenario, and scale up the obtained results with the ratio that was chosen earlier on. In addition, in the model, we can use one agent to represent a group of agents who have similar behaviors and goals. In doing so, we can efficiently model their group behavior without taking up too much unnecessary computational resources.

Chapter 5

Modelling Systems without Complete Information

5.1 Introduction

One of the difficulties in modelling multi-agent systems is to be able to represent the variability and uncertainty at a reasonable computational cost. Traditional approaches, that model individuals explicitly, do not have much success in obtaining insights into such systems in a timely manner. Taking multiple locations into consideration certainly worsens the situation. In the previous chapters, we have demonstrated that, modelling individuals according to their nature (e.g. adversarial behavior) , is a fast and effective way to tackle this problem. Depending on the quantity of agents that exist in the system and their individual impact, the approach models the systems at different levels of detail. Results show that the approach predicts the agents' performance at low computational cost and allows us to identify the characteristics that impact the system most. In addition, we have discovered that the approach is capable of modelling the systems at different levels of abstraction. In other words, we can obtain the estimated outcome of scenarios set in large terrains by modelling them in smaller terrains with similar geographical characteristics.

The proposed approach is based on the assumption that we have complete knowledge of the systems that are being modelled. Such knowledge includes, the agents' behaviors, agents' intentions and interactions, and the size and geographical characteristics of the terrain. These aspects are essential for calculating the equilibrium probabilities $q(i, c, k)$, and therefore the estimated time of events happening. The fact that we do not always have access to this information makes the modelling process more difficult, if not impossible. In this chapter, we describe a method that gains the essential information based on historical observation records.

5.2 Obtaining Motion Probabilities via Observation

Being an important component of the mathematical model, the motion probability reflects the characteristics of the terrain as well as the agents' intentions and interactions. In this section, we discuss how to estimate motion probabilities based on observation.

5.2.1 Method Description

To illustrate the idea, we use the scenario where an agent (the circle) is approaching its goal (the star) as an example (See Figure 5.1). Prior to the observation, the agent's motion in the terrain and the path that it will take to approach its goal is unknown to us (the grey area). With every move that the agent takes, we note down the previous location i and the current location j. This piece of information tells us that, in the future, should the agent occupy location i again, there is a possibility of it moving to j. Now we know that at location i, the agent has moved to location j once. This can also be interpreted as: so far, among all the neighboring locations, the agent is most likely to move to location j. Now, we have some knowledge (the white area) about the agent's motion at location i .

During the course of the observation run, we repeat the same procedure for all

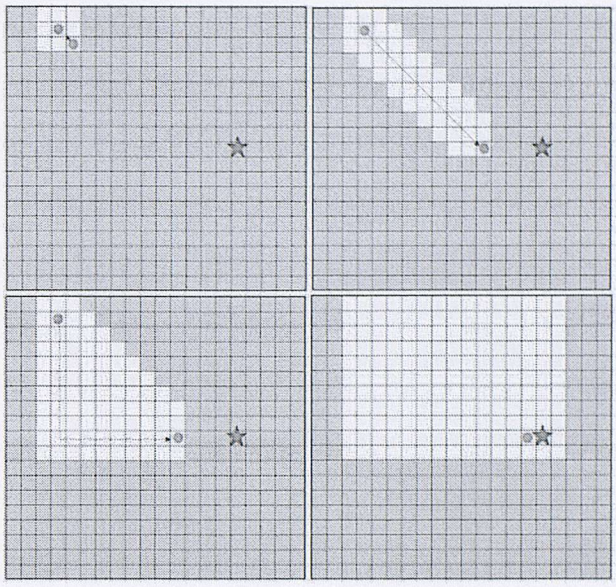


Figure 5.1: An example of deducing motion probability

locations that the agent has occupied. Our knowledge of the agent's motion increases as it traverses in the terrain (the increasing white area). At the end of the observation run, we have a record that, at any location, the number of times that the agent has moved to its neighbors. During the observation run, we have discovered one particular path from the agent's initial location to its goal. The information that is currently available to us leads to a biased view: the agent picked this path does not mean it will pick the same path in future. Therefore, we have to conduct multiple observation runs to encounter as many potential paths as possible. In doing so the bias can be greatly reduced, if not removed completely.

The motion probability $p(i, j, c, k)$ is therefore the ratio of the number of times that the agent k of team c moving from i to its neighboring location j to the total number of times that it has passed location i. The obtained motion probabilities are then simply substituted into the mathematical model (See Equation 4.11). With this plugged in, the mathematical approach is now able to model systems that agent interactions are unknown to us. Procedures of measuring motion probabilities for systems containing multiple agents are exactly the same.

During its life time, most likely an agent will not cover every location in the terrain. For locations that the agent did not occupy, we can assume that their neighboring locations share an equal probability of its arrival. Alternatively, for the agents that have destinations, we can assign the motion probabilities in such a way that, they have a high tendency of moving towards their goals. One disadvantage of this method is that it is difficult to identify locations that are occupied by obstacles: a location that was not covered during the observation runs, either has an obstacle or is simply not on the way to the agents' destinations. It is difficult to remove this defect completely. However, conducting a large number of observation runs ensures the agents exploring as many locations and paths as possible. Most likely, agents have low interests in the locations that are not occupied during the observation runs, so the their motion probabilities should have trivial impact to the estimated outcome.

5.2.2 Scenarios and Obtained Results

The described method is implemented with the help of the simulator. The agent behavior model of the simulator includes probabilistic elements, therefore, integration over many simulation runs allows us to statistically sample average agent behavior. Using the 3 adversarial agent teams scenario (See Figure 4.9) as an example, we examine the effect that incomplete system information has on the modelling process.

Equally assigning the motion probabilities to unoccupied locations allows us to identify the likelihood of an agent moving towards a particular direction while carrying out its task. For example, Figure 5.2 shows that an agent has a low probability of moving towards this direction while travelling in the terrain, for the motion probabilities of the visited locations are lower than those that are not occupied. Figure 5.3, on the other hand, is an example of a likely direction towards which the agent moves during its task. We observed that when the process is first started, the agent is in favor of this direction. As time goes on, the agent's motion is affected by the others: it starts moving towards other directions to avoid the dangers and other interactions. When the dangerous situation is

cleared, it starts moving towards its goal again. The peak in the figure suggests that, in all the simulation runs, the agent always approaches its goal from this particular direction.

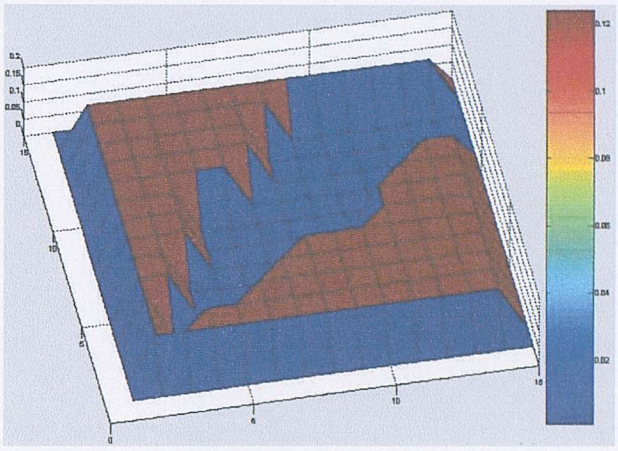


Figure 5.2: An unlikely direction in the task

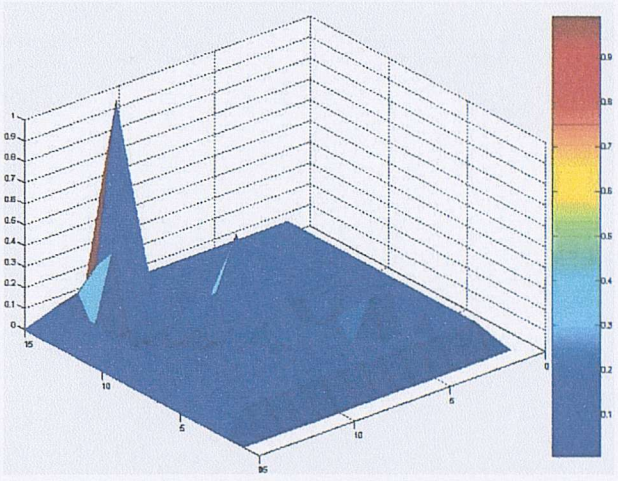


Figure 5.3: The motion probabilities of a likely direction

Substituting the obtained motion probabilities into the mathematical model (See Equation 4.11) enables us to model this scenario even though the detail of agent interactions is unknown to us. As predicted, the way that the motion probabilities are assigned for the unoccupied locations has trivial effect on the outcome (See Figure 5.4 and 5.5). Of course, when high motion probabilities are assigned to the direction of agent $1's$ goal, it arrives slightly quicker (27.41 steps) than the motion probabilities are assigned equally (29.55 steps). In both cases, it takes agent 2 around the same time to kill agent

1, 21.48 and 21.76 steps respectively. Motion probabilities of unoccupied locations have trivial impact on the time that takes agent 3 or 2 to kill each other (18.1 and 18.07 steps).

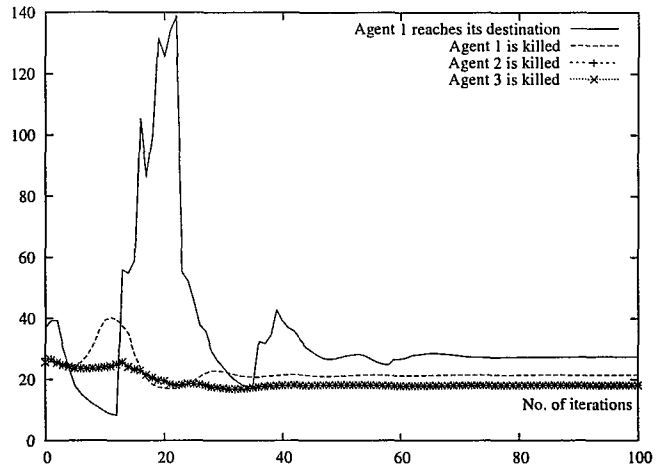


Figure 5.4: High motion probabilities are assigned to the direction of the goal

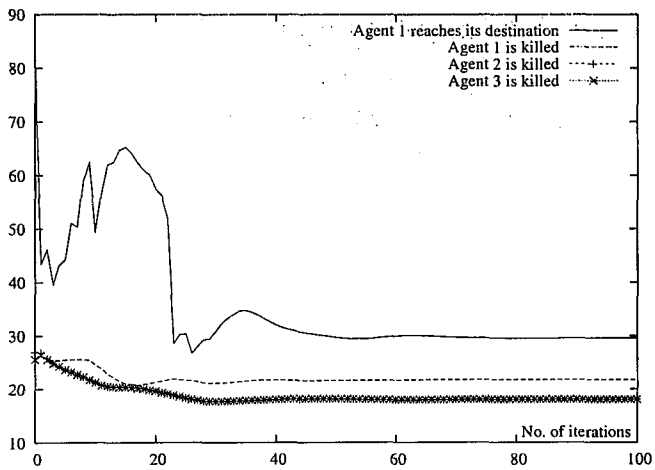


Figure 5.5: Motion probabilities of unoccupied locations are equally assigned

The results obtained from the hybrid model are comparable with the ones from the original mathematical model, where agent interactions is known to us (See Figure 4.11). As shown in Table 5.2.2, it takes agent 1 similar time to arrive its goal or get killed. A distinct discrepancy is that, for agent 2 and 3, the hybrid approach suggests a shorter survival time than the mathematical model (around 18 steps rather than 24.3 steps). This is a side effect of integrating simulation into the model. As discussed in the previous

chapter, the result of multiple simulation runs, on its own, is not sufficient in describing the agents' performance. It only represents the average time of events happening, but reveals no information on the likelihood. Another factor causing the discrepancy is the agents' shooting rate. In the simulator, an agent's shooting rate is the average rate that it launches attacks, and the exact shooting pattern is probabilistic. Hence, the outcome of events of agent 2 and 3 being killed are affected most.

Table 5.1: Results obtained for the 3 adversarial teams scenario from the three approaches

Scenarios Vs. Approaches	**Estimated Time of Events Happening**			
	Agent 2 is killed	Agent 3 is killed	Agent 1 is killed	Agent 1 reaches its goal
Simulation	445	478	299	668
Mathe.Model	24.30	24.30	18.86	26.71
Simu. and Mathe.Model(MPE)	18.07	18.07	21.76	29.55
Simu. and Mathe.Model(MPG)	18.1	18.1	21.48	27.41
*MPE: the motion probabilities are equally assigned *MPG: high motion probabilities are assigned to directions leading to the destination *The motion probabilities are calculated based on 5000 observation runs				

A potential problem of the hybrid model is that we can not detect whether a location contains obstacles. Experiments show that this aspect has little impact on the estimated outcome. To elaborate on this, we make use of the following two examples (See Figure 5.6).

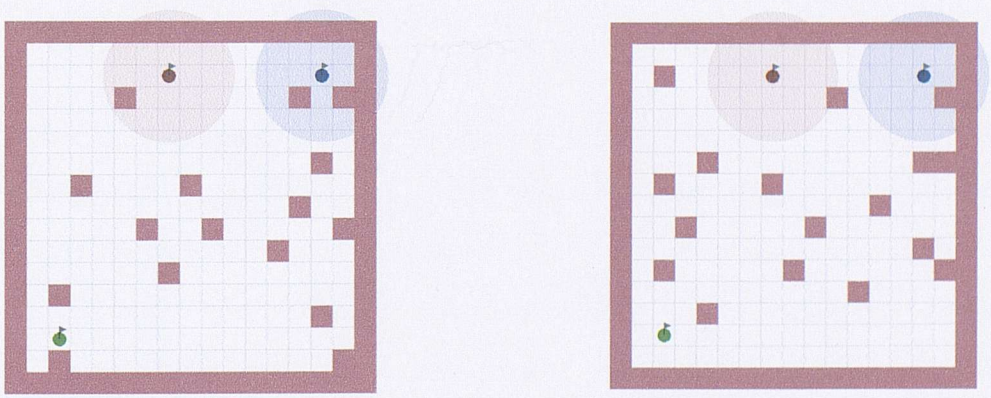


Figure 5.6: Two scenarios where obstacles exist in the terrain

For the first scenario, results show that the motion probabilities do not have much impact on the estimated time of events happening (See Figure 5.7 and 5.8) . Comparing with the result from the mathematical model (See Figure 5.9), we observe that, not knowing the locations of the obstacles, agent 1 takes much longer to reach its goal (44.14 and 42.97 steps rather than 25.92 steps). Assigning motion probabilities to unoccupied locations implies that agents may go to places that are not on their way to the destinations, and therefore it takes longer for them to arrive. In the version of the simulator that we used for the experiments, the obstacles cannot protect agents from attacks. Whereas in the mathematical model, the obstacles are shelters for avoiding attacks. Hence results from the latter suggests that agent 1 has a longer survival time and less time to arrive its goal.

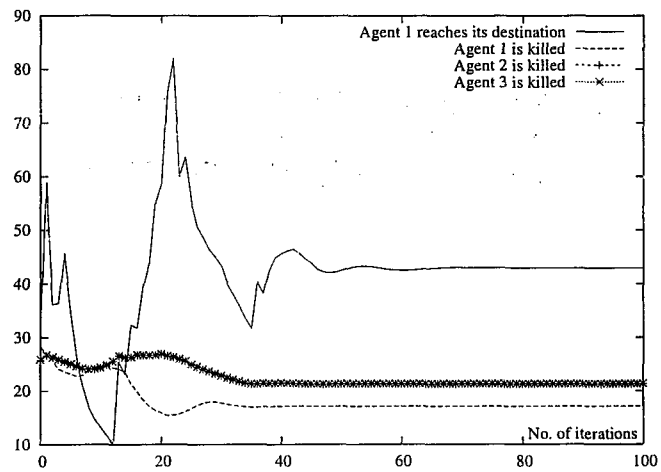


Figure 5.7: Results of scenario 1 obtained from the hybrid model (MPG)

For the second scenario, we can also observe that, for those unoccupied locations, assigning high motion probabilities to the direction of an agent's goal leads to results (See Figure 5.10) that are more inline with the mathematical model (See Figure 5.11) than assigning the motion probabilities equally (See Figure 5.12). For example, comparing with 25.36 steps in the mathematical model, agent 1 takes longer to reach its goal, 28.51 steps under MPG and 31.38 steps under MPE. The mathematical model suggests that scenarios with the two sets of obstacles have similar outcome. This is not observed in the hybrid model, which suffers from the side effect of the integration of the simulation.

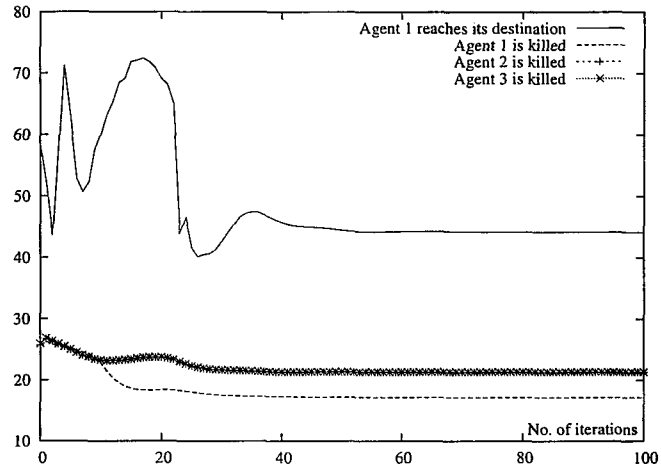


Figure 5.8: Results of scenario 1 obtained from the hybrid model (MPE)

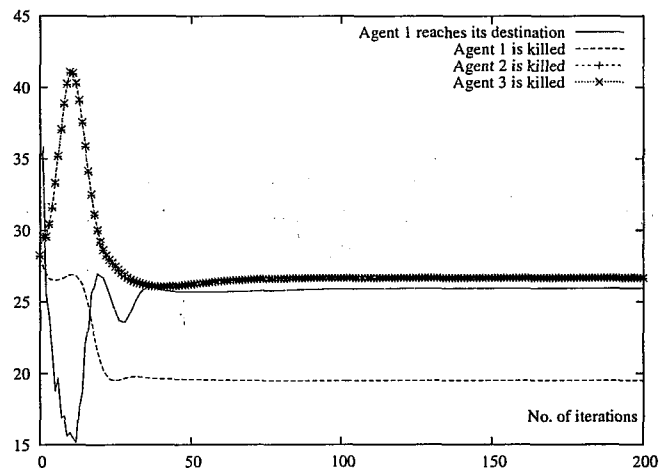


Figure 5.9: Results obtained from the mathematical model (scenario 1)

Experiments indicate that assigning high motion probabilities to directions that lead to the goal brings results that are more inline with the mathematical model. Even though integrating simulations into the mathematical model introduces some side effects, it offers valuable insights into the systems. In situations where no information on agent interactions is available to us, the hybrid model is by all means a good reference to the agents' performance.

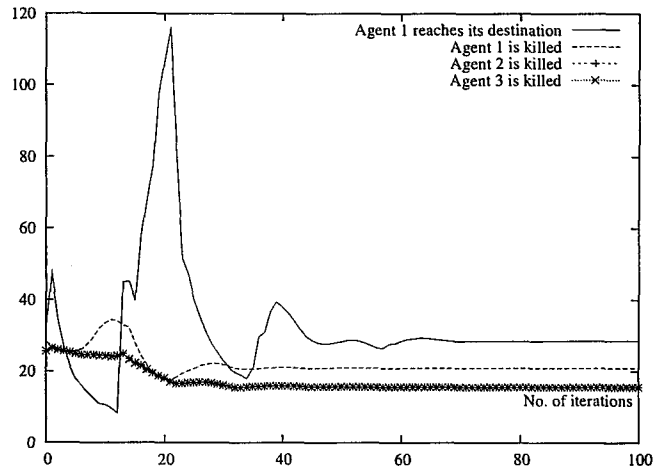


Figure 5.10: Results of scenario 2 obtained from the hybrid model (MPG)

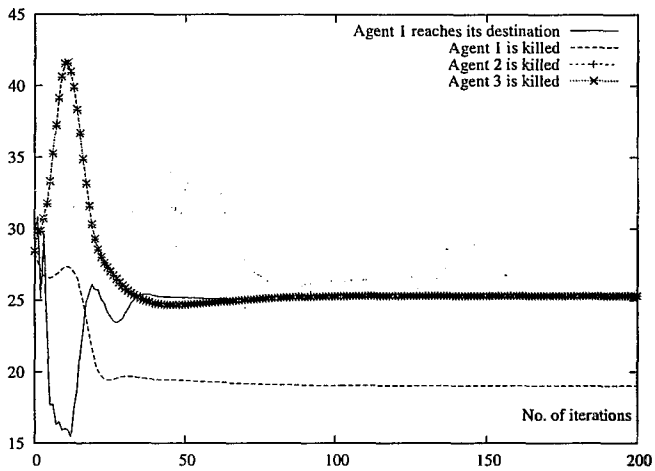


Figure 5.11: Results obtained from the mathematical model (scenario 2)

5.3 Group Detection

Apart from obtaining the motion probabilities, the observation records can also be used to detect agent groups. Of course, we only cover this aspect at a preliminary level, since plan recognition is a subject on its own.

The idea is to detect agent groups and intentions as they move in the terrain, therefore this method is based on the historical data of one observation run. Same as before, the instantaneous geographical locations of all the agents are collected at

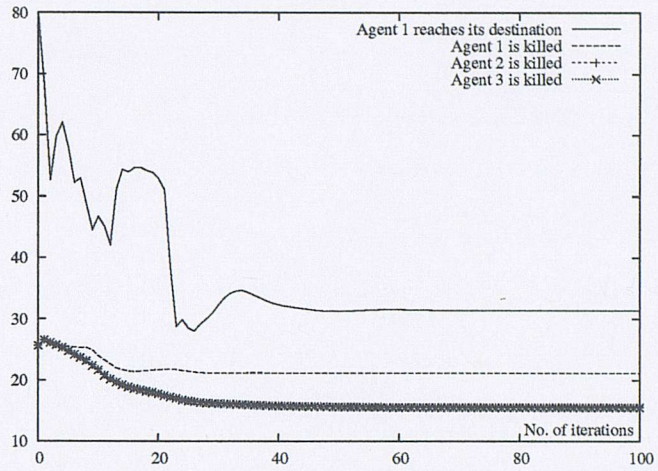


Figure 5.12: Results of scenario 2 obtained from the hybrid model (MPE)

each time step, and the directions of their motion are calculated. Similar to the way that motion probabilities are calculated, at each time step, an agent's instantaneous geographical location divides the terrain into eight sub-terrains. The sub-terrain that is in the direction of an agent's instantaneous motion is considered to be its current 'area of interest'. An 'area of interest' may be in the direction of an agent's goal or allow it to run away from its enemies. An integration of an agent's area of interest over an observation period gives us an idea that, on average, the part of the terrain that it is interested in.

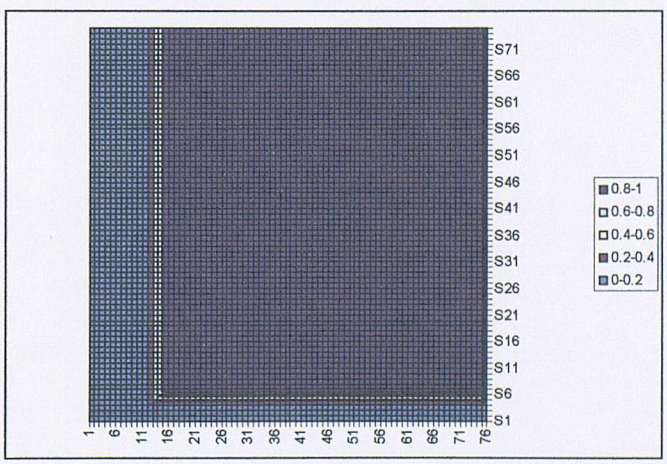


Figure 5.13: An agent's area of interest at time step 60

Figure 5.13 is an example of an agent's area of interest in the three adversarial agent

teams scenario after the observation process has started for 60 time steps. As shown in the figure, more than 80% of the time, the agent is interested in a large part of the terrain (the top right part). Agents' areas of interest usually include a big portion of the terrain and it is too general for group detection. Therefore we condense the area of interest to a point. Similar to the idea of the center of mass of an irregular shaped object, the 'point of interest' represents the point of the terrain that an agent is interested in. At the moment, the level of an agent's interest in a particular location is represented in the format of percentage, which can be treated as the weight of this location. Knowing the size of the terrain and the weight of each location, the area of interest can be shrunk to a point. Figure 5.14 and 5.15 shows the point of interest of all agents at 60 and 240 time steps respectively. At this point, based on their point of interest, the agents can easily be clustered into groups (See Figure 5.16 and 5.17).

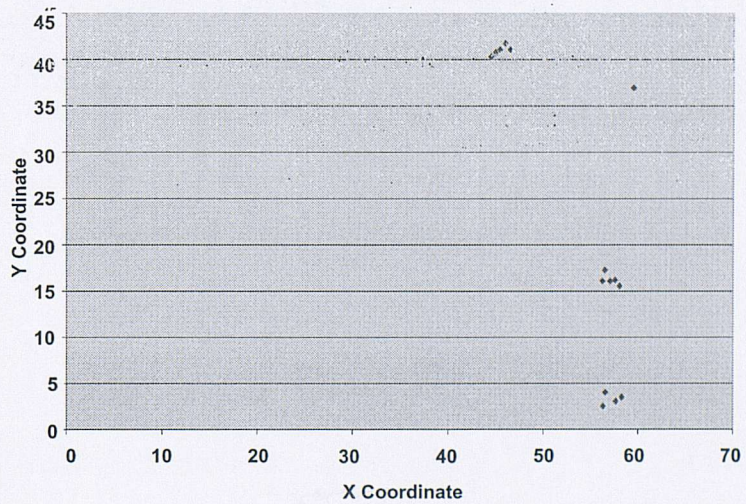


Figure 5.14: Calculated points of interests after 60 time steps

We choose K-means as the clustering algorithm, for it is a simple and easy way to partition a given data set into a pre-specified k clusters. From the data set, it randomly chooses k instances and uses them as the initial cluster centers over the clustering space. Once all the instances are assigned to its closest center, the centers are recalculated and replaced with the mean of their members. This procedure repeats until there are no further updates to the center of the clusters.

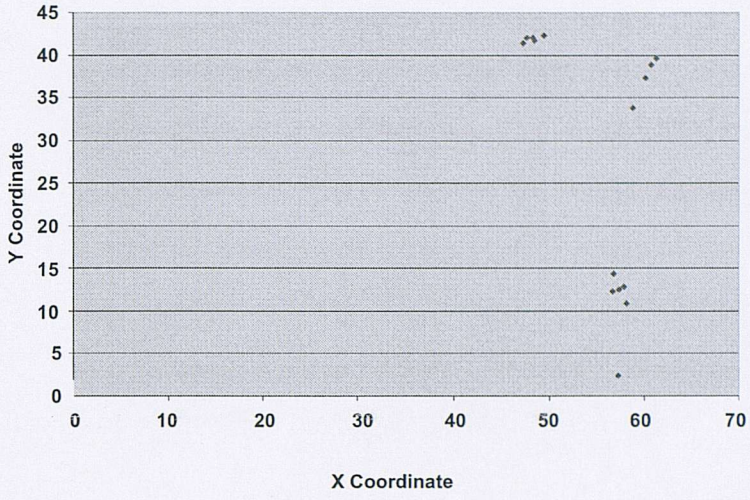


Figure 5.15: Calculated points of interests after 240 time steps

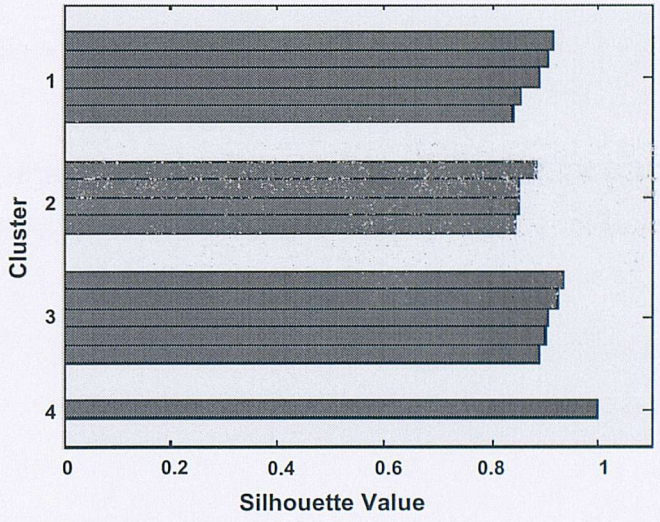


Figure 5.16: Detected agent groups after 60 time steps

K-means is known for two shortcomings: number of clusters dependency and degeneracy. Clusters dependency refers to the fact that the pre-specified value k is critical to the clustering result and it is difficult to obtain the optimal value for a given data set beforehand. Degeneracy refers to the fact that the clustering process may end with some empty and meaningless clusters. These shortcomings can be avoided by using, Y-means [52], an unsupervised clustering algorithm that is developed based on K-means

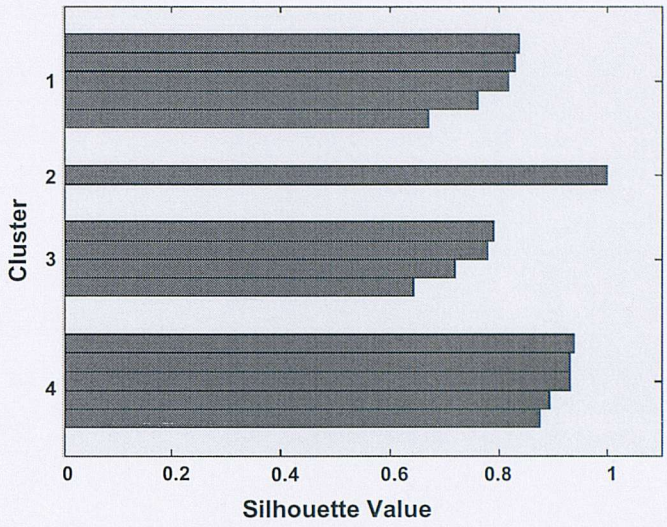


Figure 5.17: Detected agent groups after 240 time steps

and other related clustering algorithms.

Better than K-means, Y-means does not require the user to specify the number of clusters. It first partitions the data into k clusters, where k is integer between the range 1 and the total number of instances. It then searches for empty clusters and creates new clusters to replace the empty ones, if they exist. This procedure is carried out until there are no empty clusters left. In doing so, the instances that are quite different from the majority of its cluster members are removed to form new clusters where instances are more similar to each other. In addition, this procedure merges the overlapped adjacent clusters. Hence the value of k is determined automatically by splitting or merging clusters.

For this scenario, we know that the system has 3 agent groups and 5 agents in each group. However, analysis at 60 and 240 time steps, shows that 4 agent groups are detected. Observing the process as it happens, we notice that the blue team starts splitting as the red team approaches (See Figure 5.18). After 335 time steps, 1 blue agent has changed its direction dramatically to avoid encountering with the red team and the rest of the blue team continues moving towards its goal (See Figure 5.19). Experiments show that 'point of interest' offers an early detection of the agents' abnormal behaviors, such as leaving its group.

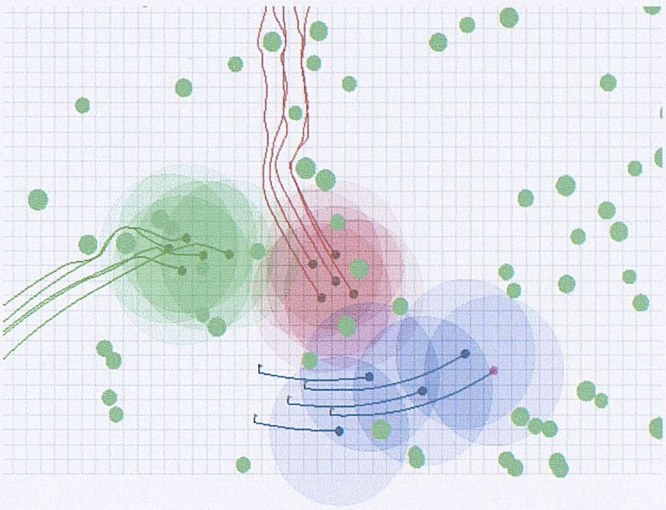


Figure 5.18: The status of the system at 60 time steps

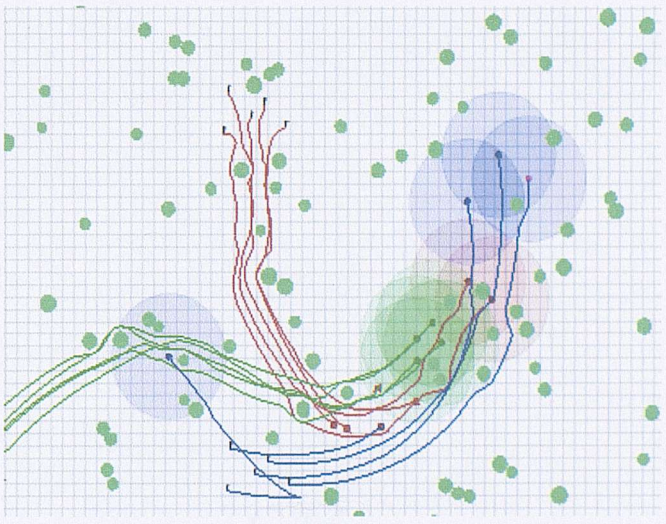


Figure 5.19: The status of the system at 496 time steps

In this chapter, we have demonstrated how to model large scale agent systems when their complete information is not available to us. The gap is filled by the historical data that are collected over a large number of observation runs. Experiments show that integrating the simulation into the mathematical model leads to minor discrepancies on the estimated outcome of events happening. However, for systems that we have no access to the essential information, the hybrid approach offers a good reference to the agents'

performance. There is a wide room for exploiting the uses of the observational method, as an example, we have demonstrated how it can be used in identifying agent groups at a preliminary level.

Chapter 6

Conclusions

6.1 Contribution

One of the well-known problems in modelling large scale agent systems is that the computational complexity of the system increases exponentially with respect to the number of agents that are being modelled. Traditional techniques, such as building differential equations and simulation, model the agents separately. How to deal with the high computational complexity caused by the large number of agents has always been a difficult issue. In this thesis, we presented a novel approach that detaches the relation between the computational complexity of a system and the number of agents. The approach, based on G-Networks, an extension to the traditional queueing networks, models the agents according to their nature (e.g. type or adversarial behavior). In doing so, the computational complexity of a system depends on the agent types and the number of locations that are being modelled. This chapter summarizes our contributions and discusses the potential extension to our approach.

In this work, we divide the systems into two categories: systems that contain a large population of entities (indistinguishable agents) and each entity has negligible impacts on the system, and systems that contain a small number of entities (distinguishable agents) and each entity has distinct impact on the system. A human body can be considered as

an instance of the former type in the field of biology. The latter is commonly seen in military planning where a key civilian or a group of civilians is the focus of the study. Despite the great variance in the quantity and the impact of the agents, these systems have one thing in common: the agents collaborate and compete to achieve their goal. Depending on the number of the agents that are of interest, the approach models the systems at different levels of detail.

In Chapter 3, we have demonstrated how to model the systems with a large number of indistinguishable agents taking actions in a single location. The scenarios that we used are commonly seen in biology, for example, keeping the viruses under control or assessing a drug's performance in cancer treatment. The sheer quantity of the agents makes it impractical to model them separately. We therefore examine the interactions between the agent types and focus our study on how such interactions affect the density of the agents. The approach allows us to identify the parameters that affect the system most as well as the settings that lead to the desired outcome (e.g. keeping the density of the viruses under a certain level). It can also be used to identify the boundary within which the models have steady state solutions.

In Chapter 4, we have presented models for the systems with distinguishable agents taking actions in a terrain with multiple locations. We used military planning scenarios to illustrate how to build the models given the agent interactions. Because the distinguishable agents have a strong individual impact on the system, we examine the interactions between agents rather than agent types. With the approach, we can not only obtain the outcome of a mission under a certain setting but also identify the technical characteristics of an agent that affect the outcome most. Comparing different strategies allows us to work out a plan that achieves the desired outcome in the most cost efficient manner.

By modelling the same scenarios and comparing the obtained results, we validated the mathematical approach against the simulator [43,44] that was developed in our group.

Designed with different objectives in mind, the simulator offers a visual representation of the underlying processes of a system whereas the mathematical approach examines the system at a higher level of abstraction and provides an estimated numerical outcome of the system. As a result of the difference between the two approaches, minor discrepancies are detected among the results. While analyzing the results obtained from the mathematical model, we have discovered that the proposed approach is capable of modelling the systems at different levels of abstraction, where we obtain an estimated outcome of scenarios set in a large space by modelling them in a much smaller one. This property further reduces the computational complexity of a system.

The models described in Chapter 3 and 4 are built based on the assumption that the interactions between the agents or agent types are known to us. This assumption restricts the number of systems that can be modelled, since large scale systems are known for their uncertainties. In Chapter 5, we have presented an extension to the model, which fills the information gap with historical observation data, i.e. integrating simulation in the mathematical approach. Results show that information, such as the average agent behavior over many simulation runs, has insignificant impact on the estimated outcomes. During the experiments, we have also discovered that the approach may potentially be used for group detection. Of course, group detection is a subject on its own and we can simply use an example to illustrate a potential extension to the approach.

To conclude the thesis, based on the established mapping between the G-Networks parameters and agent simulations, we proposed a stochastic approach that provides performance estimates at a timely manner with low computational complexities. The approach is limited to study systems that settle down to a steady state. In Chapter 3, for example, the systems do not reach steady states under all parameter settings (See Figure 3.4, 3.5). In those cases, the proposed approach is not able to provide performance estimates. We also limit ourselves to time invariant systems where the parameters do not depend on time, although the system varies with time. The system of equations may be very large or infinite (e.g. the example in Section 3.3.3), we therefore limit our analysis to seeking

steady-state solutions. We confine our approach to deal with systems that agent behaviors are known to us. Because we model the systems at an abstract level, we neglect actions that are taken place inside the agents.

6.2 Potential Extensions

In this work, the obstacles in the system are not agent dependent. That is to say, all the agents share the same obstacles. This assumption does not affect the precision of the models in our case because the entities that we model are similar in size. However the obstacles are not always agent independent. For example, a building might be an obstacle for a car but not a pedestrian. It confines us from modelling the scenarios where the entities vary too much in size, for the obstacles may have a strong impact to the entities' behaviors. A potential extension to the approach would be to establish different sets of obstacles according to the sizes of the entities. In our examples, the geographical characteristics of the locations are the same throughout the terrain. With the approach, we can also explore how different terrains, e.g. rivers and mountains, affect a mission's outcome.

During the project, we have designed simulations where agents only have information of their surroundings within a certain range. In such scenarios, the police sticks to a pre-assigned patrol route when there are no civilians or terrorists in his neighborhood. The police puts his tasks on hold when he is needed for protecting a civilian or catching a terrorist and resumes the tasks once the situation is cleared. Another type of interesting scenarios is the case where an terrorist's identity is not revealed unless he launches an attack. Our approach can not model these scenarios, for we assumed that the agents are fully aware of the others. It is worth investigating whether the approach can be modified so that such scenarios can be modelled.

In Chapter 3 and 4, we have presented two ways of constructing models based on the population size. The scenarios in these two chapters may seem unrelated: we used

biological scenarios, e.g. cancer treatment, to illustrate how to build models for systems with indistinguishable agents and military planning in an urban environment for systems with distinguishable agents. It is in fact possible to combine these two types of models so that more complicated scenarios can be studied. In the peace-keeping of a city, for example, the police forces can not attack the crowd that are involved in a riot, instead, they use teargas to disperse the crowd and prevent them from approaching a certain direction. The gas itself does no harm to the civilians, but its existence prevents them from achieving their goals. On the other hand, the terrorists may want to use chemical weapons to harm the civilians. In other cases, the police may possess no attacking ability themselves. To help the civilians reaching their goals, the police may travel with them and release some chemical substances that slow the terrorists down.

For these scenarios, we are interested in how the chemical weapons affect the behaviors of the civilians or the performance of the police. The density of the chemical substances can be easily modelled, as illustrated in Chapter 3. Such densities can then be integrated in the models built for the distinguishable agents. In the case of the teargas, for example, its density can be considered as a factor that contributes towards an agent's motion probability in a certain direction. In other cases, the density of the chemical weapon can be treated as a type of adversarial action. Integrating the two systems in such a way allows us to model the emerging trend of the peace keeping in the urban environment. The challenge would be how to model the degradation of the chemical weapons over time.

Another method that is commonly used by the terrorists is the *suicide bomber*. Depending on the strength of the bomb, suicide bombers can harm or even kill many people in one go as well as set the buildings on fire or even destroy them. Depending on the speed of the rescue team, the fire might also spread and get out of control. Scenarios where a single agent kills a number of agents in one unit of time are not modelled by our approach. However, these scenarios can be modelled by exploring the 'batch removal' function provided by the G-Networks. Again, the challenge lies in how to model the

spread and degradation of fire over time.

When modelling urban scenarios, we assumed that, in one unit of time, all the agents can only move to one of their adjacent neighbors. This implies that all agents travel at the same speed in the terrain. In some scenarios, we may encounter special agents that travel much faster than the others. The contrast in speed can be reflected in the model by allowing agents move to locations that are more than one unit away. That is to say, an agent's speed can be altered by changing its motion probabilities. This allows us to model agents of different technical levels, e.g systems containing the civilian, the normal police and the special forces.

Bibliography

[1] Load balancing. Retrieved from World Wide Web http://www.webopedia.com/TERM/l/load_balancing.html on 21st Aug 2004.

[2] Load balancing: Where the action is. Retrieved from World Wide Web http://www.networkmagazine.com/article/NMG20000426S0014 on 21st Aug 2004.

[3] Mathematical modelling. Retrieved from World Wide Web http://www.fact-index.com/m/ma/mathematical_model.html on 21st June 2004.

[4] Mathematics awareness week: Mathematics and decision making, 1996. Retrieved from World Wide Web http://math.ucsd.edu/~williams/mathaw/mathaw.html on 25th March 2004.

[5] Plus magazine - features: Agner krarup erlang, 1997. Retrieved from World Wide Web http://plus.maths.org/issue2/erlang/ on 22nd June 2004.

[6] The earliest known uses of some of the words of mathematics, 1999. Retrieved from World Wide Web http://mail.mcjh.kl.edu.tw/~chenkwn/mathword/q.html on 22nd June 2004.

[7] Agentlink report, 2003. Retrieved from World Wide Web http://www.AgentLink.org on 31st March 2004.

[8] Network load balancing technical overview, 2004. Retrieved from World Wide Web http://www.microsoft.com/windows2000/techinfo/howitworks/cluster/nlb.asp on 21st Aug 2004.

[9] K. Amin and A. Mikler. Dynamic agent population in agent-based distance vector routing. In *2nd International Workshop on Intelligent Systems Design and Applications (ISDA)*, 2002.

[10] G. R. Bitran and R. Morabito. Open queueing networks: Optimization and performance evaluation models for discrete manufacturing systems. *Production and Operations Management*, 5(2):163–193, 1996.

[11] P. P. Bocharov and V. M. Vishnevskii. G-networks: Development of the theory of multiplicative networks. *Autom. Remote Control*, 64(5):714–739, 2003.

[12] S. K. Bose. Introduction to queueing theory, 2004. Retrieved from World Wide Web `http://home.iitk.ac.in/~skb/ee679/ee679.html` on 29th Aug 2004.

[13] Jeffrey M. Bradshaw. An introduction to software agents. In Jeffrey M. Bradshaw, editor, *Software Agents*, pages 3–46. AAAI Press / The MIT Press, 1997.

[14] J. C. Brustoloni. Autonomous agents: Characterization and requirements, 1991. Report CMU-CS-91-204.

[15] B. Burmeister. Models and methodology for agent-oriented analysis and design. In *the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems*, 1996.

[16] Liu Chengqing, Marcelo H. Ang, Hariharan Krishnan, and Ser Yong Lim. Virtual obstacle concept for local-minimum recovery in potential-field based navigation. In *ICRA*, pages 983–988. IEEE, IEEE, 2000.

[17] R. B. Cooper. *Introduction to Queueing Theory.* The Macmillian Company, 1981.

[18] R. B. Cooper. Introduction to queueing theory, 1981. Retrieved from World Wide Web `http://www.cse.fau.edu/~bob/publications/IntroToQueueingTheory_Cooper.pdf` on 29th Aug 2004.

[19] Luiz Marcio Cysneiros and Eric S. K. Yu. Requirements engineering for large-scale multi-agent systems. In Alessandro F. Garcia, Carlos José Pereira de Lucena, Franco

Zambonelli, Andrea Omicini, and Jaelson Castro, editors, *SELMAS*, volume 2603 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2002.

[20] A. K. Erlang. The theory of probabilities and telephone conversations. *Nyt Tidsskrift for Matematik B*, 20, 1909.

[21] Oren Etzioni and Daniel S. Weld. Intelligent agents on the internet: Fact, fiction, and forecast. *IEEE Expert*, 10(3):44–49, 1995.

[22] A. Ferrier. Queueing theory, 2004. Retrieved from World Wide Web `http://www.new-destiny.co.uk/andrew/past_work/queueing_theory/Andy/` on 29th Aug 2004.

[23] P.M. Fiorini. Queueing theory, 2004. Retrieved from World Wide Web `http://www.cs.usm.maine.edu/~pfiorini/erlang-B-C-models.pdf` on 29th Aug 2004.

[24] J. M. Fourneau, L. Kloul, and D. Verchère. Multiple class g-networks with list oriented deletions. *European Journal of Operations Research*, 126(2):250–272, 2000.

[25] Jean-Michel Fourneau and Erol Gelenbe. Flow equivalence and stochastic equivalence in g-networks. *Computational Management Science*, 1(2):179–192, 2004.

[26] Jean-Michel Fourneau, Erol Gelenbe, and Rina Suros. G-networks with multiple classes of negative and positive customers. *Theor. Comput. Sci.*, 155(1):141–156, 1996.

[27] Jean-Michel Fourneau and Dominique Verchère. G-networks with synchronized partial flushing, 1995. Technical Report.

[28] J.M. Fourneau, L. Kloul, and F. Quessette. Multiple class g-networks with jumps back to zero. In *MASCOTS '95: Proceedings of the 3rd International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 28–32, Washington, DC, USA, 1995. IEEE Computer Society.

[29] S. Franklin. *Artificial Mind.* MIT Press, Cambridge, MA, USA, 1995.

[30] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *ECAI '96: Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, pages 21–35, London, UK, 1997. Springer-Verlag.

[31] E. Gelenbe and A. Labed. G-networks with multiple classes of signals and positive customers. *European Journal of Operations Research*, 108(2):293–305, 1998.

[32] Erol Gelenbe. Random neural networks with positive and negative signals and product form solution. *Neural Computation*, 1(4):502–510, 1989.

[33] Erol Gelenbe. Stability of the random neural network model. *Neural Computation*, 2(2):239–247, 1990.

[34] Erol Gelenbe. Product form queueing networks with positive and negative customers. *Journal of Applied Probability*, 28:656–663, 1991.

[35] Erol Gelenbe. G-networks: A unifying model for neural nets and queueing networks. In Herbert D. Schwetman, Jean C. Walrand, Kallol Kumar Bagchi, and Doug DeGroot, editors, *MASCOTS*, pages 3–8. The Society for Computer Simulation, 1993.

[36] Erol Gelenbe. G-networks with instantaneous customer movement. *Journal of Applied Probability*, 30(3):742–748, 1993.

[37] Erol Gelenbe. G-networks with signals and batch removal. *Probability in the Engineering and Information Sciences*, 7:335–342, 1993.

[38] Erol Gelenbe. Learning in the recurrent random neural network. *Neural Computation*, 5(1):154–164, 1993.

[39] Erol Gelenbe. G-networks: Multiple classes of positive customers, signals, and product form results. In Mariacarla Calzarossa and Salvatore Tucci, editors, *Performance*, volume 2459 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2002.

[40] Erol Gelenbe and Jean-Michel Fourneau. G-networks with resets. *Perform. Eval.*, 49(1/4):179–191, 2002.

[41] Erol Gelenbe, P. Glynn, and K. Sigman. Queues with negative arrivals. *Journal of Applied Probability*, 28:245–250, 1991.

[42] Erol Gelenbe, Peter G. Harrison, Edwige Pitel, Onno J. Boxma, and Jean-Michel Fourneau. G-networks - new queueing models with additional control capabilities (panel). In *SIGMETRICS*, pages 58–59, 1995.

[43] Erol Gelenbe, Khaled Hussain, and Varol Kaptan. Enabling simulation with augmented reality. In Mariacarla Calzarossa and Erol Gelenbe, editors, *MASCOTS Tutorials*, volume 2965 of *Lecture Notes in Computer Science*, pages 290–310. Springer, 2003.

[44] Erol Gelenbe, Varol Kaptan, and Khaled Hussain. Simulating the navigation and control of autonomous agents. In Per Svensson and Johan Schubert, editors, *Proceedings of the Seventh International Conference on Information Fusion*, volume I, pages 183–189, Mountain View, CA, Jun 2004. International Society of Information Fusion.

[45] Erol Gelenbe, Varol Kaptan, and Yu Wang. Biological metaphors for agent behavior. In Cevdet Aykanat, Tugrul Dayar, and Ibrahim Korpeoglu, editors, *International Symposium on Computer and Information Sciences*, volume 3280 of *Lecture Notes in Computer Science*, pages 667–675. Springer, 2004.

[46] Erol Gelenbe, Varol Kaptan, and Yu Wang. Simulation and modelling of adversarial games. In *6th European GAME-ON Conference on Simulation and AI in Computer Games*, pages 40–44, 2005.

[47] Erol Gelenbe and Guy Pujolle. *Introduction to queueing networks.* Chichester : J. Wiley, 1998.

[48] Erol Gelenbe and M. Schassberger. Stability of product form g-networks. *Probability in the Engineering and Information Sciences*, 6:271–276, 1992.

[49] Erol Gelenbe and Yu Wang. A trade-off between agility and resilience. In *13th Turkish Symposium on Artificial Intelligence and Neural Networks*, pages 209–217, 2004.

[50] Erol Gelenbe and Yu Wang. A mathematical approach for mission planning and rehearsal. In Raja Suresh, editor, *Defense Transformation and Network-Centric Systems*, volume 6249 of *Proc. SPIE*, pages 197–207, May 2006.

[51] Erol Gelenbe and Yu Wang. Modelling large scale autonomous systems. In *The 9th International Conference on Information Fusion*, pages 183–189, 2006.

[52] Y. Guan, A. Ghorbani, and N. Belacel. Y-means: A clustering method for intrusion detection. In *the Canadian Conference on Electrical and Computer Engineering*, volume a2, pages 1083– 1086, 2003.

[53] B. Hayes-Roth. An architecture for adaptive intelligent systems. *Artificial Intelligence*, 72(1-2):329–365, 1995.

[54] Thomas D. Haynes and Roger L. Wainwright. A simulation of adaptive agents in a hostile environment. In *SAC '95: Proceedings of the 1995 ACM symposium on Applied computing*, pages 318–323, New York, NY, USA, 1995. ACM Press.

[55] W. Henderson. Queueing networks with negative customers and negative queue lengths. *Journal of Applied Probability*, 30:931–942, 1993.

[56] T. Herman. Introduction to queueing theory, 2004. Retrieved from World Wide Web `http://nest.cs.uiowa.edu/22C178f00/pdf/~qintro.pdf` on 29th Aug 2004.

[57] C. Hewitt. Viewing control structures as patterns of passing messages. *Artificial Intelligence*, 8(3):323–364, 1977.

[58] M. Hlynka. Queueing history, 2004. Retrieved from World Wide Web `http://www2.uwindsor.ca/~hlynka/queue.html` on 22nd June 2004.

[59] S. Hohmann, J. Nielsen, and H. Kitano. Yeast systems biology - concepts, 2004. Retrieved from World Wide Web `http://www.symbio.jst.go.jp/symbio/yeast/paper/WhitePaperYSB.pdf` on 21st June 2004.

[60] G. Huang, A. Abur, and W. K. Tsai. A multi-level graded-precision model of large scale power systems for fast parallel computation. *Mathematical computing modelling*, 11:325–330, 1998.

[61] Y. Huang. Basic queueing theory m/m/* queues introduction, 2004. Retrieved from World Wide Web `http://www.cs.gmu.edu/~huangyih/756/queueing.pdf` on 29th Aug 2004.

[62] Carlos Iglesias, Mercedes Garrijo, and José Gonzalez. A survey of agent-oriented methodologies. In Jörg Müller, Munindar P. Singh, and Anand S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555, pages 317–330. Springer-Verlag: Heidelberg, Germany, 1999.

[63] Alain Jean-Marie. Qmips. Retrieved from World Wide Web `http://www-sop.inria.fr/mistral/infos/QMIPS.html` on 21st June 2004.

[64] N. R. Jennings and M. J. Wooldridge. *Agent technology: foundations, applications, and markets.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[65] Shaler Stidham Jr. Analysis, design, and control of queueing systems. *Oper. Res.*, 50(1):197–216, 2002.

[66] Varol Kaptan. Modeling autonomous agents in military simulations. *Ph.D. Dissertation, Fall 2006, University of Central Florida*, 2006.

[67] R. Keen and J. D Spain. *Computer Simulation in Biology: A BASIC Introduction.* Wiley-Liss Press, New York, NY, USA, 1992.

[68] David George Kendall, 1998. Retrieved from World Wide Web `http://www-history.mcs.st-andrews.ac.uk/Mathematicians/Kendall.html` on 22nd June 2004.

[69] S. Kim. Biological system simulation: A literature survey, 2003. Retrieved from World Wide Web `http://people.cs.vt.edu/~haebang/coursework/ssimul/research/Kim.pdf` on 21st June 2004.

[70] David Kinny, Michael Georgeff, and Anand Rao. A methodology and modelling technique for systems of BDI agents. In Rudy van Hoe, editor, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.

[71] L. Kleinrock. *Queueing systems Volume 1: Theory.* Wiley-Interscience, 1975.

[72] John Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[73] R. H Lathrop. Knowledge-based avoidance of drug-resistant hiv mutants, 1999. Retrieved from World Wide Web `http://www.findarticles.com/p/articles/mi_m2483/is_1_20/ai_54367770/print` on 21st Aug 2004.

[74] D.V. Lindley. The theory of queues with a single server. In *PROCEEDINGS of the Cambridge Philosophical Society*, pages 227–289, 1952.

[75] Z. Liu, M.H.Ang, and W.K.G.Seah. A potential field based approach for multi-robot tracking of multiple moving targets. In *Environment and Management International Conference*, 2003. Retrieved from World Wide Web `http://guppy.mpe.nus.edu.sg/~liuzheng/paper/HNICEM03.pdf`.

[76] Pattie Maes. Artificial life meets entertainment: lifelike autonomous agents. *Commun. ACM*, 38(11):108–114, 1995.

[77] A. A. Markov. Extension of the limit theorems of probability theory to a sum of variables connected in a chain. *The Notes of the Imperial Academy of Sciences of St. Petersburg VIII Series*, XXII(9):552–577, 1907.

[78] X. Meng. Introduction to queueing theory, 1999. Retrieved from World Wide Web `http://www.cs.panam.edu/~meng/Course/CS6354/Notes/meng/master/node4.html` on 29th Aug 2004.

[79] Martin A. Nowak and Karl Sigmund. Evolutionary dynamics of biological games. *Science*, 303(5659):793–799, February 2004.

[80] H. S. Nwana and D. T. Ndumu. A brief introduction to software agent technology. In Jennings and Wooldridge [64], pages 29–47.

[81] R. O. Onvural. Survey of closed queueing networks with blocking. *ACM Computing Surveys*, 22(2):83–121, 1990.

[82] A. Pivk and M. Gams. Intelligent agents in e-commerce. *Electrotechnical Review*, 67(5):251–260, 2000.

[83] J. Reif and H. Wang. Social potential fields: A distributed behavioral control for autonomous robots. In *International Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 431–459, 1995.

[84] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and V. Lorensen. *Object-Oriented Modelling and Design.* Prentice-Hall, 1991.

[85] Stuart Russell and Peter Norving. *Artificial Intelligence: A Modern Approach.* Prentice Hall, Upper Saddle River, New Jersey, USA, 1995.

[86] T. L. Saaty. *Elements of queueing theory, with applications.* McGraw-Hill, 1961.

[87] Yoav Shoham. An overview of agent-oriented programming. In Jeffrey M. Bradshaw, editor, *Software agents*, pages 271–290. MIT Press, Cambridge, MA, USA, 1997.

[88] David Canfield Smith, Allen Cypher, and Jim Spohrer. Kidsim: programming agents without a programming language. *Commun. ACM*, 37(7):54–67, 1994.

[89] Martin Spengler and Bernt Schiele. Multi-object tracking: Explicit knowledge representation and implementation for complexity reduction. *Cognitive Vision Workshop 2002, September 2002*, 2002.

[90] P. Stubbs. Introduction to queueing theory, 2004. Retrieved from World Wide Web `http://www.philstubbs.com/erl_extr.doc` on 29th Aug 2004.

[91] T. Sunsted. An introduction to agents, 1998. Retrieved from World Wide Web `http://www.javaworld.com/javaworld/jw-06-1998/jw-06-howto_p.html` on 31st March 2004.

[92] Yu Wang. G-networks and the modeling of adversarial agents. In *16th International Conference on Artificial Neural Networks*, pages 330–339, 2006.

[93] Yu Wang. Numerical modelling of autonomous agent movement and conflict. *Journal Computational Management Science*, 3(3):207–223, 2006.

[94] F. Wattenberg. Prolog for a winter day, 1996. Retrieved from World Wide Web `http://www.math.montana.edu/frankw/ccp/modeling/simple/winter/learn.htm` on 25th June 2004.

[95] G. Weiss. *Multiagent systems : a modern approach to distributed artificial intelligence.* MIT Press, Cambridge, MA, 1999.

[96] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115--152, 1995.

[97] Michael J. Wooldridge and Nicholas R. Jennings. Agent theories, architectures, and languages: A survey. In Michael J. Wooldridge and Nicholas R. Jennings, editors, *Workshop on Agent Theories, Architectures & Languages (ECAI'94)*, volume 890 of *Lecture Notes in Artificial Intelligence*, pages 1–22, Amsterdam, The Netherlands, January 1995. Springer-Verlag.

[98] J. Xu. Introduction to queueing theory, 2004. Retrieved from World Wide Web `http://clayton.ncat.edu/comp690/` on 29th Aug 2004.

[99] X. P. Yun and K. C. Tan. A wall-following method for escaping local minima in potential field based motion planning. In *8th International Conference on Advanced Robotics*, pages 421–426, 1997.