Formation and organisation in robot swarms.

OTHMAN, Wan Amir Fuad Wajdi.

Mrst Jur venire uny uampus
_Sheffield S1 1WB

§H@ffi§ld Hallsm University
k§§fflln0 and IT Services
Ad§§tt»Centre City Campus
ihgffield S1 1WB

REFERENCE

Formation and Organisation in Robot Swarms

Wan Amir Fuad Wajdi Othman

A thesis submitted in partial fulfilment of the requirements of
Sheffield Hallam University
for the degree of Doctor of Philosophy

March 2009

Declaration

This is to certify that I am responsible for the work submitted in the thesis, that the original work is my own except as specified in acknowledgements, and that neither the thesis nor the original work contained therein has been submitted to this or any institution for a higher degree.

Signature: .....

Name : **WAN AMIR FUAD WAJDI OTHMAN**

Date : **10 March 2009**

Acknowledgement

I would like to express my sincere gratitude to my research supervisor, Bala P. Amavasai, for his excellent guidance and consistent support during my research. I am grateful to him for giving me an invaluable opportunity to work on four different aspect of swarming over the past four years. Thanks for broadening my scope and helping me achieve my goals. I would like to communicate my appreciation to Jon R. Travis, Bala P. Amavasai and Reza Saatchi for their interest and enthusiasm, to read, modify and comment on the manuscript.

I would like to thank all colleagues of the Microsystems and Machine Vision Laboratory, particularly Jan Wedekind, Kim Chuan Lim, Tan Kang Song, Manuel Boissenin and Arul Selvan for their support and encouragement. Also, I would like to extend my gratitude to Fabio Caparrelli and Stephen McKibbin for many insightful suggestions, fruitful discussions and constructive comments.

I would like also to thank Alan Goude for his enthusiasm in teaching and guidance on my extra curriculum lecture. I benefited a lot from his valuable course on Object Oriented Programming.

Thanks to the Ministry of Higher Education of Malaysia and University Science of Malaysia for giving me the opportunity to pursue my research, and for sponsoring my studies. Special thanks to members of the Registrar office of USM particularly to Fatimah Othman and Ramli Osman, for their kind support and cooperation towards the completion of this thesis.

Finally, but most importantly, I would like to express my sincere love and appreciation to my mother, Faizah Othman, my wife, Fazrin Mohamad Azemi and my son Aleef Imran for their unconditional love, support, patience, encouragement and du'a throughout the work which has taken many hours that should have been dedicated exclusively to them. No matter what happens they have always stood behind me and have shared in my happiness and difficulties. Thanks again to my wife, she has tolerated all my stress, helping me with her unbelievable love. Also I would like to express my love to my late father, brothers and sisters as well as parents in law for their continuous support and du'a.

Abstract

A swarm is defined as a large and independent collection of heterogeneous or homogeneous agents operating in a common environment and seemingly acting in a coherent and coordinated manner. Swarm architectures promote decentralisation and self-organisation which often leads to emergent behaviour. The emergent behaviour of the swarm results from the interactions of the swarm with its environment (or fellow agents), but not as a direct result of design. The creation of artificially simulated swarms or practical robot swarms has become an interesting topic of research in the last decade. Even though many studies have been undertaken using a practical approach to swarm construction, there are still many problems need to be addressed. Such problems include the problem of how to control very simple agents to form patterns; the problem of how an attractor will affect flocking behaviour; and the problem of bridging formation of multiple agents in connecting multiple locations. The central goal of this thesis is to develop early novel theories and algorithms to support swarm robots in pattern formation tasks. To achieve this, appropriate tools for understanding how to model, design and control individual units have to be developed. This thesis consists of three independent pieces of research work that address the problem of pattern formation of robot swarms in both a centralised and a decentralised way.

The first research contribution proposes algorithms of line formation and cluster formation in a decentralised way for relatively simple homogenous agents with very little memory, limited sensing capabilities and processing power. This research utilises the Finite State Machine approach.

In the second research contribution, by extending Wilensky's (1999) work on flocking, three different movement models are modelled by changing the maximum viewing angle each agent possesses during the course of changing its direction. An object which releases an artificial potential field is then introduced in the centre of the arena and the behaviours of the collective movement model are studied.

The third research contribution studies the complex formation of agents in a task that requires a formation of agents between two locations. This novel research proposes the use of L-Systems that are evolved using genetic algorithms so that more complex pattern formations can be represented and achieved. Agents will need to have the ability to interpret short strings of rules that form the basic DNA of the formation.

Table of Contents

Figure Index

Index of Tables

Chapter 1 Thesis Overview

1.1 Motivation

Since the dawn of time, humans have observed a variety of interesting and intriguing patterns found in nature due to the natural movements of animals and insects. A flock of birds in formation in the sky, a school of fish which turns and flees in perfect coordination (Shaw 1962), a group of eusocial insects (*e.g.* ants) foraging cooperatively for food. These kind of behaviours that lead to organised formation is termed as "swarm behaviour" (Liu & Passino 2000). In recent times, researchers from many diverse fields have converged to study the interaction in biological swarms and how to model them, through the observation of organisation and evolution in the swarm agents. Researchers in the applied sciences, for instance, have shown an even greater interest in swarm behaviour since the understanding of these behaviours can lead to new optimisation techniques such as the Particle Swarm Optimisation (Kennedy & Eberhart 1995) and Ant Colony Optimisation (Bonabeau *et al.* 1999). These behaviour inspired algorithms can be applied in many fields, such as in networks and telecommunication systems (Bonabeau *et al.* 1999), robotics (Beni 2005, Cao *et al.* 1997) *etc.*

Recent advances in robotics in general and electronics in particular have started to make the deployment of large numbers of inexpensive agents or robots for many practical applications more feasible. Such applications include for example search and rescue type tasks where these inexpensive agents are tasked with looking for survivors in collapsed buildings after a natural disaster like the aftermath of an earthquake. Agents in this instance have to perform dangerous or explorative tasks in hazardous, unknown and remote environments. In deploying these

agents, the number of autonomous agents involved can be very large, ranging from hundreds to thousands.

When dealing with large numbers of agents, many problems need to be addressed. Such problems include the agent's design and architecture, task allocations, control strategy, localisation and so forth. Another important question that needs to be addressed is one of organisation. Agents in the system should be able to form and organise themselves around complex patterns which are generally required to perform specific tasks in a complex arena. This thesis focuses on the latter problem, *i.e.* the organisation of the robot swarms.

Although many approaches and solutions have been proposed to address the organisation issues, as swarm robotics is relatively a new field, there are still many aspects that can be investigated. With a fuller understanding, researchers may find solutions that lead to better algorithms.

In this thesis, the wide range of techniques and algorithms currently being developed or available is examined and studied in-depth. With this new understanding this thesis

- proposes algorithms of line formation and cluster formation for relatively simple multi-robot system using existing state based model,
- studies the impact of collective movement model behaviours in the presence of an attractor unit (artificial potential field), and
- proposes a novel method of robots formation connecting two locations by using Lindenmayer Systems in conjunction with evolutionary algorithms.

Original contributions are offered in the three key areas above through the study and analysis of existing algorithms, improvements of these algorithms and finally and most importantly the contribution of new algorithms.

In this Chapter, a brief introduction to self organisation, research context and outline of the thesis is presented. Firstly an overview of self organisation systems including some definitions

is presented and some of the main ingredients that make up these swarm systems are explained. The research in this thesis is then put into context. Finally, the outline of this thesis is presented in Section 1.4.

1.2 Self Organisation

Self organisation in swarm systems refers to a broad range of pattern-formation processes in nature. These include sand grains forming rippled dunes (Figure 1.1), orderly rows of clouds in the sky, flocking behaviours in birds and so on. Camazine *et al.* (2001) in their book (p.8) provided an "open" definition on self organisation as reproduced below:

> *"Self-organisation is a process in which patterns at the global level of a system emerge solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern".*

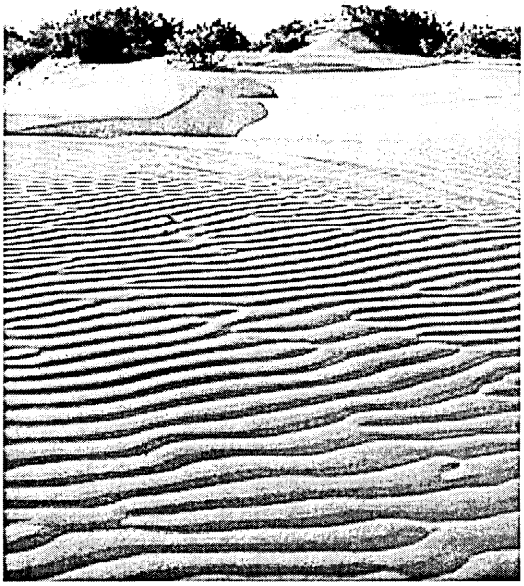


Figure 1.1: An example of pattern formation in nature, showing sand dunes.

Bonabeau *et al.* (1997) who worked closely on insects gave another definition on self organisation which focuses more on ethological aspect as follows:

> *"Self organisation does not rely on individual complexity to account for complex spatiotemporal features that emerge at the colony level, but rather assumes that interactions amongst simple individuals can produce highly structured collective behaviours".*

This thesis proposes the following summary definition

> *"In self-organised swarming systems, pattern formation usually occurs through local interactions of agents in the system without intervention by external directing influences".*

There are four basic ingredients that may contribute to the self-organising systems:

- multiple interactions
- amplification of fluctuation and randomness
- positive feedback
- negative feedback

Interaction is the main ingredient and it is a basic requirement for self-organising systems. In nature, interaction is needed to allow an agent to obtain information which is used to determine a response. Obtaining information from an interaction is a result of some kind of communication with the nearest neighbours or its environment. In the simplest case of flocking birds for example, the local information acquired in the interaction is simply the relative position of other birds in the neighbourhood. This information is gathered directly without the need of direct communication, *e.g.*. bird to bird communication. It is also unnecessary for birds to leave some sort of "marker" in the environment to communicate with the others in the flock. In the case of foraging ants, ants also do not require direct communication with other individuals. However, ants leave behind in their tracks a type of chemical substance called pheromones as an environment marker to communicate with other ants (Bonabeau *et al.* 1997). In cases like these (ants foraging and birds flocking), it demonstrates that only by having

indirect communication during an interaction is sufficient to produce complex behaviour. In many other cases, information usually transfers by direct communication. A well known example of this type of interaction is the dancing performance by some species of bees. When a bee returns from foraging to the hive, the bee will perform a dance that conveys information about the approximate location of the nectar source (Bonabeau *et al.* 1999).

In biological systems, random fluctuation is a common ingredient in boosting up self organisation performance. Many of these systems do actually rely on certain stochastic elements to some degree for behavioural flexibility. The amplification of fluctuation and randomness often leads to the discovery of new solutions. Moreover, these fluctuations will also act as seeds in which new solutions and structures can grow. A popular example of the random fluctuation is caused by stochastic trail following in ant colonies. In the beginning, the ants will follow trails imperfectly due to the low concentrations of pheromone on the ground (Deneubourg 1983). But when an ant loses the trail and is lost in the environment, this ant has the potential to find an undiscovered food source. The newly found food could be a better food source than the currently being utilised by the colony. From this example, it shows that random fluctuations are also vital to the swarm systems.

Another common ingredient in self-organising systems is positive feedback or cumulative causation. Positive feedback promotes radical changes in the system by reinforcing it in the same direction. A commonly observed example of positive feedback can be found again in the trail-laying in ants. When an ant finds food, it will leave behind a pheromone trail while returning to the nest. Others who find this trail will follow the trail to the food source, and they will reinforce the initial trail as they return to the nest. As the result of positive feedback, the more ants that use the trail, the stronger the pheromone concentration will be.

Negative feedback in the self organising systems acts as a balancing mechanism of the effect of positive feedback. In nature, the autocatalytic process usually requires an opposing force in most cases, otherwise the system will use huge amounts of resources for a single particular activity. Negative feedback usually occurs due to the depletion of limited individual or

resources. In swarm systems, the negative feedback can be in the form of saturation, exhaustion overcrowding, and even competition within individuals. In the case of foraging ants, the negative feedback will come from the exhaustion of the food, overcrowding at the food source, competition between two or more food sources, limited number of available ants, and so on and so forth.

As there is no unique or satisfactory definition of self organisation, the summary above serves as a set of heuristic rules to design or discover a self organising system.

In systems that lack self organisation, order or organisation can be imposed on them in many different ways. The order not only can come through the presence of a supervisory team but through various directives such as pre-existing patterns in the environment as well.

1.3 Context

The origin of the work that is presented in this thesis is intimately related to and emerges from the development of the I-SWARM project (Intelligent Small World Autonomous Robots for Micro-manipulation), funded by European Union Information Society Technologies (IST) 6th framework programme (FP6-IST project 507006) which began in January 2004 and completed at the end of 2008 (Woern *et al.* 2006). The project aims to take a leap forward in robotics research by bringing together experts and combining expertise in micro-robotics, in distributed and adaptive systems, and in self-organising biological swarm systems. The project also seeks to produce technological advances to facilitate the mass production of micro-robots, which can then be employed as a "real" swarm consisting of up to 1000 robot agents. The agents that form the swarm will each be equipped with limited pre-rational on-board intelligence. The swarm will consist of a huge number of heterogeneous robots, differing in the type of sensors, manipulators and computational power. Such a robot swarm is expected to perform a variety of applications in the not too distant future, including micro assembly, biological, medical or cleaning tasks.

The main challenge in the I-SWARM project is to build a micro robot with the initial aim of achieving a size of $4mm^3$ (2mm x 2mm x 1mm) complete with sensors and manipulators. The work in the first phase was a joint effort of the partners in the Consortium to define the minimum capabilities of a micro robots which are able to self organise and able to have emergent behaviours. As the size is the main issue in the project, based on the robot hardware conceived, the behaviour of the robot is designed by using a bottom-up approach.

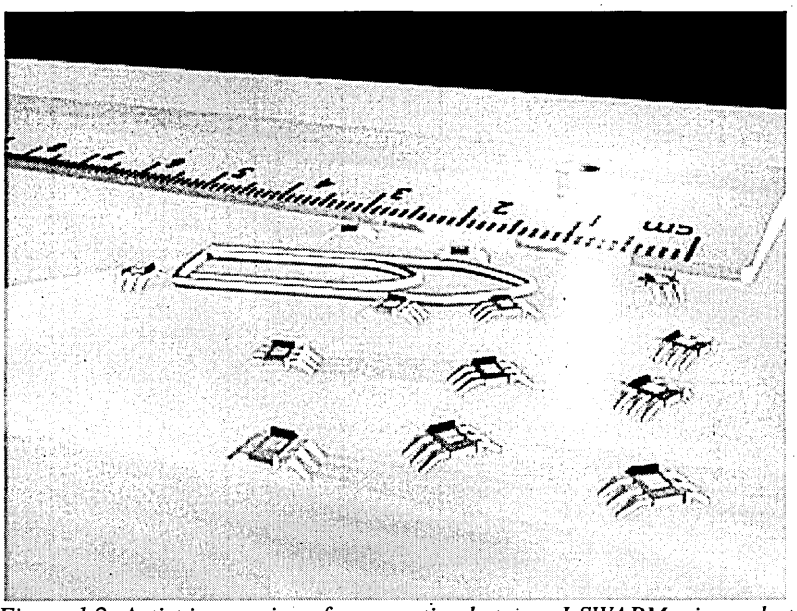


Figure 1.2: Artist impression of cooperation between I-SWARM microrobots

Each of the I-SWARM robot has three "legs" (two at the front and one at the back) made from special materials (electro-active polymers) as a locomotion unit. As the swarm needs a continuous supply of energy, micro solar cells and a thin film battery that acts as a buffer have been mounted onto the main platform. The onboard electronics consists of an 8051 micro controller core, analogue circuitry (for the power drivers for the actuators), and the A/D converters (for communication and sensor modules and power management). Optical communication using custom fabricated infra-red LEDs and photodiodes technology has been chosen as the communication mod for the I-SWARM robots. The communication range is set to about 2-4 times the size of the robot in four directions (front, right, back and left relative to the robot).

At the time of writing, the I-SWARM robots have been manufactured as shown in Figure 1.3, with the final size of $27mm^3$ (3mm x 3mm x 3mm). The remaining tasks now are to program the robots so that they can exhibit some sort of intelligence and show some kind of emergence behaviours.

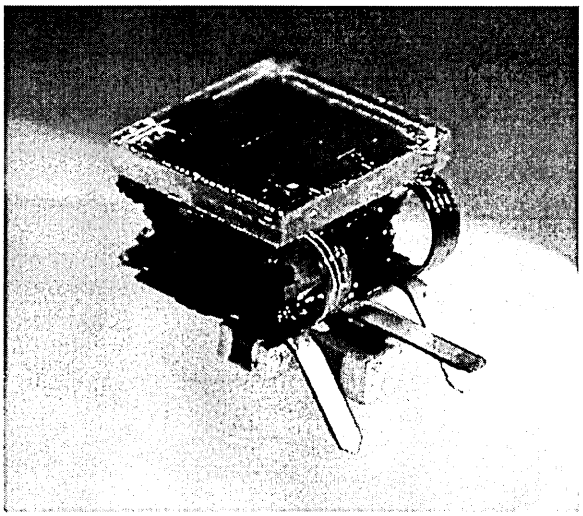


Figure 1.3: The final I-SWARM robot with dimension of 3x3x3 mm^3.

1.4 Structure of the Thesis

This rest of the thesis is organised as follows:

- Chapter 2 gives background information and provides a literature review surrounding other research related to this thesis. These include some background studies in the biological, artificial intelligence and robotics field. An overview of the current state of the art in the field is also presented.

- Chapter 3 introduces two robot swarm control algorithms which are used for distributed pattern formations. In these control algorithms, there is no explicit communication between agents and the pattern formations are formed based solely on reactivity of the agents towards its environments. Agents in this study have very little memory, limited sensor capabilities and processing power.

- Chapter 4 models collective movements or aggregations of robots swarms using simple flocking rules. From the model, aggregation behaviours that emerge from the different movement models of relatively simple agents, which differ only in the maximum turning angle and sensing range, are examined.

- Chapter 5 presents an original contribution on complex pattern formations of robots swarms by combining Lindenmayer Systems (or L-Systems in short) and genetic algorithms. In this study, it is shown that the pattern that is formed when connecting two locations can be achieved and represented using simply evolved L-Systems, provided each robot has the ability to interpret short strings of L-Systems that form the basic DNA of the formation.

- Finally in Chapter 6, conclusions are drawn. The results and the performance of the algorithms are discussed. Additionally the contributions of the research are summarised and recommendations and directions for possible future research are proposed.

Chapter 2 Literature Survey

2.1 Pattern Formations

Since the dawn of time, humans have been fascinated with the regular natural patterns that emerge around them – social insects foraging, birds flocking, shepherding, not to mention countless examples from physical systems such as the orderly rows of clouds and the washboard pattern of sand ripples in deserts.

In biological systems, groups of the same species of animals seem to move as a single unit, changing direction in a split second which has led some researchers to believe that some kind of communication or even "thought transference" must be involved as argued by Parrish & Edelstein-Keshet (1999). In reality this behaviour is less mysterious.

Many believe that birds must have leaders, *e.g.* the bird at the front of the flock leads and the others follow. But, in fact, most bird flocks do not have a leader at all. There is no overall control. Instead, the flock movements are determined by the instantaneous decisions of individual birds.

Birds follow simple rules in response to interactions with their neighbours in the flock. Orderly flock patterns arise from these simple rules, reacting to the movements of its neighbours. None of the birds have a sense of the overall flock pattern. The flock is coordinated without a coordinator and organised without an organiser.

There are many reasons to believe why animals aggregate in numbers. The most common reason seems to be that it serves as a defence against predators. Having many eyes together ensures that at least some will spot a predator while the others are feeding, resting, or looking in the opposite directions (Vabø & Nøttestad 1997)(Howard 1929). Parrish and Edelstein-Keshet (1999) pointed out that aggregation is actually an evolutionarily advantageous state: where it is believed that aggregation may increase the chances of survival of newborns and juveniles from being killed by predators, such that the reproduction of the species can be continued. Secondly, the aggregation also helps in the search for food; where a large number of individuals has more capability to *sense* and *search* than a single one.

In 1975, Powell conducted experiments on bird aggregation where he took a number of Starlings (a species of bird) and put them in an aviary. He then separated some of the birds on their own and some in a group of around ten. He made an artificial hawk and flew it over the Starlings and noticed that birds on their own took a longer time to react than in the groups. He concluded that even though it might be advantageous in some aspect for the Starlings to forage on their own, it is better for them to forage as a group and take turns in looking out for predators as they will be able to react more quickly in danger.

A number of the anti-predator strategies in schools of fish, such as *split*, *join* and *vacuole* (Figure 2.1), performed by schools during predator attack are some of the most interesting behaviours in a swarm (Vabø & Nøttestad 1997). Another benefit of moving in formation is the dilution effect. The dilution effect is simply that the bigger the group size, the smaller the probability that each individual is attacked. Krause (1994) stressed that odd individuals are attacked first; however that does not mean that each individual is fighting to gain access to the safest location in the swarm. In 1994, Cresswell observed and studied the behaviour of a species of bird called Redshanks. He found that once the group of the birds reaches a certain number, vigilance no longer has such a crucial effect on the group. He also realised that it actually became harder for an individual to be singled out by a predator for attack and some times by staying together it would even deter a predator.

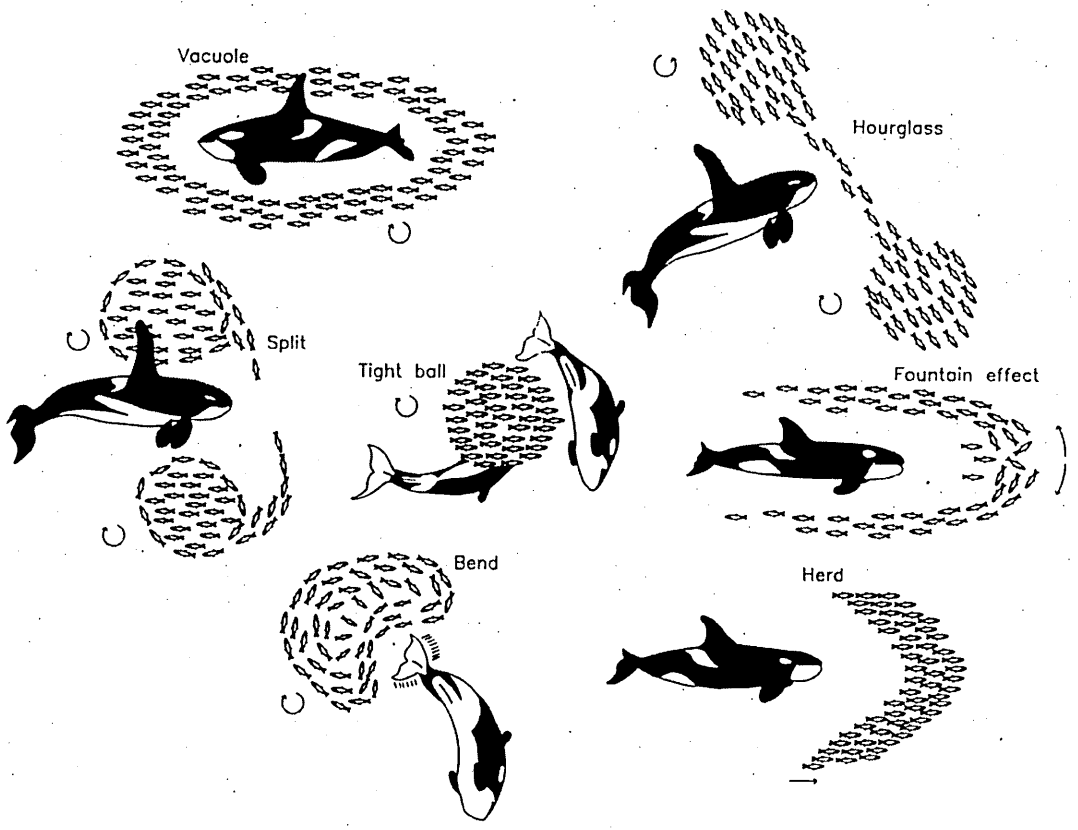


Figure 2.1: Schematic presentation of several anti-predator strategies in a school of fish. (Taken from Vabø & Nøttestad 1997)

In the special case of flying in a V-formation by large birds such as geese and pelicans, there is an energy benefit (Lissaman & Shollenberger 1970), since following birds can take advantage of vortexes in the air produced by the ones ahead of them (Gould & Heppner 1974). Although such formations clearly have leaders, these are temporary ones. Because a leading bird does not gain any energetic advantage from its position, it will drop back after some time while another takes the lead. It is not known if flock members do this on a rotation basis, although it is possible that larger and stronger birds are in the lead a greater percentage of the time. Alternatively, the V-form may reflect a mechanism by which birds avoid collisions with other birds and stay in visual contact all the time (Gould & Heppner 1974). Additional background on biological swarms and why they aggregate can be found in (Parrish *et al.* 2002)(Hamilton 1971) and references there in.

2.1.1 Pattern Forming Paradigms

In designing artificial swarms, a variety of approaches have been proposed to create global behaviour or pattern formation of a group of mobile robots. Spears *et al.* (2005a) divided the approaches in to two significant paradigms; Biomimetics and Physicomimetics.

2.1.1.1 Biomimetics

Biomimetics is a general description for an engineering process or system that mimics (imitating, copying, or learning from) biological systems. The term emerged from biochemistry and applies to an infinite range of chemical and mechanical phenomena, from cellular processes to whole-organism functions. As an early example, the Wright brothers are said to have built their aeroplane structure based on observations and analysis of bird flight. However, researchers diverge in precisely how to define biomimetics. "Biomimetics" is often a vague term, much like the "intelligent" term.

In the field of swarm engineering, Reynolds was one of the first researcher to investigate behavioural control animation (1987). He developed a system to model flocking characteristics of birds and fishes. It was based on three dimensional computational geometry of the sort normally used in computer animation or computer aided design. He called the generic simulated flocking creatures as boids. The basic flocking model consists of three simple steering behaviours which describe how an individual boid manoeuvres based on the positions and velocities to its nearby flockmates. More detail about Reynolds's flocking algorithm will be described later on in the next section.

Based on the schooling behaviour of a group of tuna, Hanada *et al.* (2007) proposed an adaptive flocking algorithm. In this algorithm, an agent first dynamically selects two of the neighbouring agents within its perception range and maintains a uniform distance with them, resulting in three neighbouring agents form a regular triangle. As the number of agents grow, the group of agents will form an equilateral triangle lattice. Secondly, in the presence of obstacles, the swarm of agents is required to be divided into multiple smaller group in order to avoid the obstacles. The split takes place by the relative degree of attractive force termed

favourite force, which is similar to Newton's law of Universal Gravitation, that helps agents to decide their direction in various environmental conditions. Based on the magnitude of favourite vector $\vec{f}$, each agent decides where to move. A favourite vector $\vec{f}_j$ for the passageway s_j is defined by $|\vec{f}_j| = w_j/d_j^2$. where w_j is the width of the passageway and d_j is the distance to the passageway, as shown in the Figure 2.2. A set of favourite vectors $\{\vec{f}_j | 1 \le j \le n\}$ is the representation of the passageways, and the agent will select the maximum magnitude of $\vec{f}_j$ denotes by $|\vec{f}_j|_{max}$. By combining the above methods, the swarm agents are enable to split into multiple groups, and also can rejoin as a big group according to the environmental conditions.

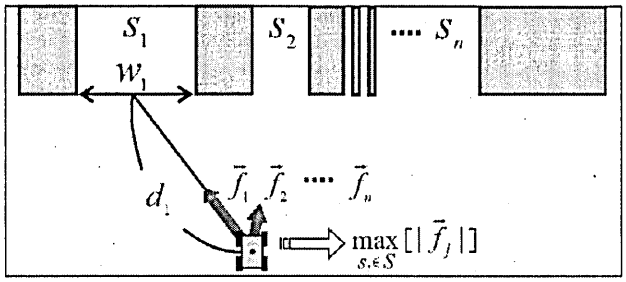


Figure 2.2: Illustration of a direction decision according to an environment computation of magnitudes for each favourite vector. (Taken from Hanada et al. 2007).

Another recent example of research in this category is the "pherobots" or pheromone robot developed by Payton *et al.* (2004). Pherobots mimic chemical pheromones released by insects to produce sophisticated organised group activity that emerges out of the simple interactions between individuals. The key concept of pherobots is "Virtual Pheromones" which provide a diffusive local-neighbourhood interaction mechanism by which the robots communicate and coordinate. Unlike chemical pheromones released by insects in the environment, virtual pheromones are tied to the robots themselves. In addition, virtual pheromones are propagated as symbolic messages and are received only by nearby neighbours. More detail about pherobots will be described later on in the next section (Swarm Robotics section).

Bayazit *et al.* (2002) proposed using rule-based roadmaps to achieve better group behaviour. In this technique, first a roadmap as in Figure 2.3 is built. A roadmap is simply a connectivity graph encoding representation of feasible paths in the environment. Each node of the graph is a configuration of the robot that satisfies certain requirements, collision-free for instance. Connections between nodes of the roadmap graph represent feasible paths. Secondly, rules at each node are added. The rules may be as simple as "Go to next node in your path"; or can be as complex as "wait for others to arrive, then select a leader, follow the leader". Their results show that the the performance of agents using rule-based roadmap behaviours is very close as if the agents have complete global knowledge of the arena.

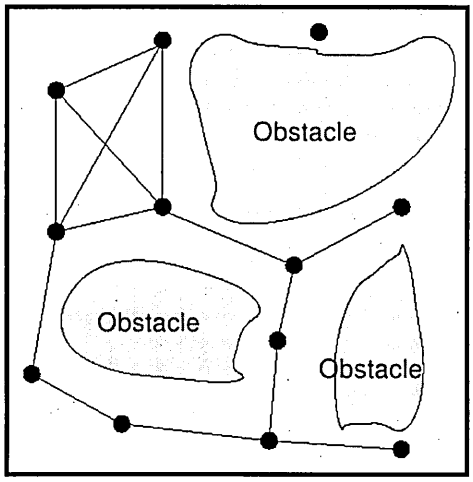


Figure 2.3: A roadmap. Black dots represent nodes; connections between nodes represent feasible paths. (taken from Bayazit et al. 2002)

Bayazit *et al.* (2004) then extended their model to achieve different behaviours from their swarm robots. One of the interesting behaviours presented is shepherding between a dog and a flock of sheep. The dog agent tries to move the flock toward a goal, the dog steers the flock from the rear and if any subgroup separates out, it is the dog's job to move the subgroup back to the flock. Their work showed that complex group behaviours can be generated if some global information of the environment is available which can not be modelled with local information alone.

In the European SWARM-BOTS project, the agents behaviours are directly inspired by the collective behaviour of social insects colonies and other social animal societies (Dorigo *et al.* 2004a). In particular the project focuses on the study of the mechanisms which govern the processes of self-organisation and self-assembling in artificial autonomous agents.

2.1.1.2 Physicomimetics

Another approach to creating global behaviour of a group of mobile robots is called "physicomimetics" or "artificial physics" (Spears *et al.* 2005a). Physicomimetics is a general description for engineering processes or systems which gain inspiration from physical systems such as fluid flow analyses, Newtonian analyses and kinetic analyses. The key points in physicomimetics are:

- Any aggregate behaviour seen in classical physics is potentially reproducible with collections of mobile robots.
- Any design is not restricted to copying physical systems precisely, *i.e.* modifications can be made.
- Understanding of classical physics can be used to synthesise the emerged collective behaviour.

In physicomimetics, the research is focused on robotics behaviours that are similar to those shown by solids, liquids and gases (Spears *et al.* 2005a). In solids, crystalline formation for example, is excellent for distributed sensing tasks, to create a virtual antennae or synthetic aperture radar. For such tasks it is important to maintain connectivity and a lattice geometry.

Liquids are good for obstacle avoidance or narrow passage traversal tasks, while moving towards a goal, since fluids easily manoeuvre around obstacles while retaining connectivity. Gases are useful for coverage, sweeping and exploration. For these tasks it is necessary that coverage can be maintained, even if with individual robot failures. Gas-like behaviours are created using purely repulsive forces.

Cheng *et al.* (2005) proposed an algorithm for coordinating a swarm of homogenous mobile agents to spatially self-aggregate into arbitrary shape using only local interactions, which they called SHAPEBUGS. SHAPEBUGS consists of two main processes; trilateration and gas expansion movement. A trilateration process allows an agent to find its perceived position on the consensus coordinate system, and subsequently adjust it; while gas expansion movement model will force agents to disperse within the defined 2-D shape. The advantage of the algorithms are that; agents can easily aggregate into any user-specified shapes, using a formation process that is independent of the number of agents within formation; and secondly agents can automatically adapt to increase and decrease of agents, as well as accidental displacement.

Zarzhitsky *et al.* (2005) introduced a chemical plume tracing (CPT) method based on computational fluid dynamics. The algorithm itself is divided into three subtasks; starting from finding the chemical, then tracing it to the source using CPT method, and finally identifying the source. In finding the chemical, agent uses a method called *casting*, which consists of zigzag or spiralling motion to increase exploration coverage. In tracing the plume, first, the agents use gravitational forces (artificial physics) to arrange themselves into a hexagonal formation and form a mobile adaptive sensor network, so that agents could share real flow-field parameters of fluid dynamics with six of their closest neighbours. These flow-field parameters or variables are use to calculate the next navigational decision using the proposed technique called *fluxotaxis*. Fluxotaxis uses the concept of mass flux, which can be written in a differential equation form as: $-\frac{\partial \rho}{\partial t} = \nabla \cdot (\rho \vec{V})$ where ρ is the mass density of the plume, $\vec{V}$ is the fluid's velocity, and the product of $\rho \vec{V}$ is called the mass flux, or the rate of change of mass flow per unit area. With fluxotaxis, each agent in the robotic lattice computes the amount of local chemical flux $\rho \vec{V}$, passing through virtual surfaces formed by neighbouring swarm agent. In addition, fluxotaxis is designed to maximise the use of available sensor data by combining the fluid velocity and chemical velocity (Spears *et al.* 2005b). The final subtask is to

identify the source of the chemical. They (Zarzhitsky *et al.* 2005) showed that their fluxotaxis algorithm is able to demonstrate statistically and practically significant gains in performance over other two most popular alternatives, *i.e.* chemotaxis and anemotaxis, even in an environment with obstacles. Even though their current results look promising, they have yet to include a more advanced turbulence model, learning the threshold of the plume / chemical, and increasing the number of obstacles.

By using what is described as *social potentials* techniques, Balch and Hybinette (2000) achieved large scale multi-agent formations. The technique was inspired by the crystal generation process. Each agent had local attachment sites attracted to other agents. When the swarm encounters obstacle, agents are able to avoid obstacle depending on the behaviour based rule combining the concept of an attractive and repulsive forces; *i.e.* repulsion from obstacles with attraction to the goal. The technique seems easy to implement however, the parameters need effort to adjust to perform successful flocking.

Spears and Gordon (1999) showed how to control swarm robot systems using a physicomimetics framework. Their initial application on solids based pattern formation, required that a swarm of micro-air vehicles (MAVs) self organise into a hexagonal lattice, creating a distributed sensing grid with a fixed spacing between MAVs (Kellogg *et al.* 2002). In liquids-based formation, they use the same approach as solids-based pattern formation only by changing the parameter that balances the attractive and repulsive components (Gordon-Spears & Spears 2002). The switch between the two behaviours (solid and liquid) acts very much like phase transition. In gas-based formations which are good for sweeping the arena, swarm robots must not only avoid obstacles but they must also sweep behind the obstacles to minimise holes in the coverage. In this case, the swarm robots must not move too quickly since it may cause a failure to sweep behind the obstacle, and they must not to move too slow. To achieve this, the optimum speed of the swarm robots has to be found (Spears & Gordon 1999).

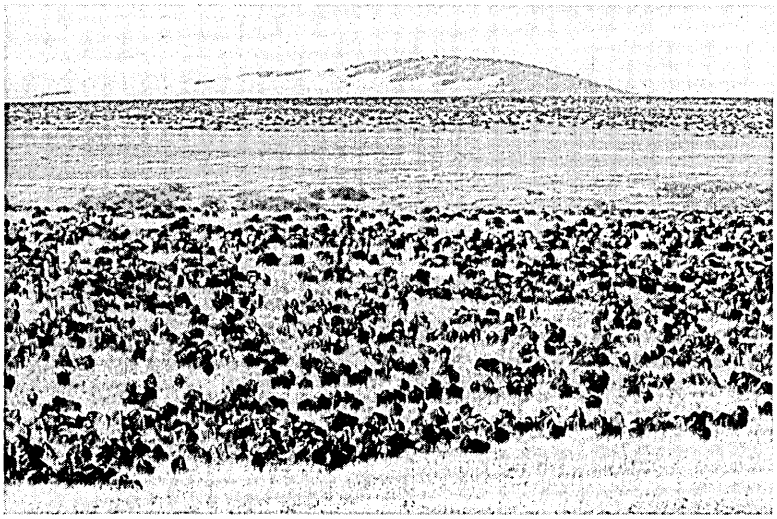
2.1.2 Organised Formations

Organised formations problem in a group of robots can be described as the coordination of the robots to form and maintain a formation of a certain shape, such as forming a line (Bahceci *et al.* 2003). Solutions for this problem are currently being used in search and rescue operations, space explorations and remote terrain, landmine removal, unmanned aerial vehicles (UAVs), control of satellites *etc.*

Various animal species also exhibit organised formation of patterns as a result of collective and cooperative behaviours amongst individual. Couzin and Krause (2003) state that, organised formations occurred when each entity in a group maintains a specific distance and orientation to each other while in motion. Examples of such organised formations include birds flocking, fish schooling and wildebeests migrating as shown in Figure 2.4.

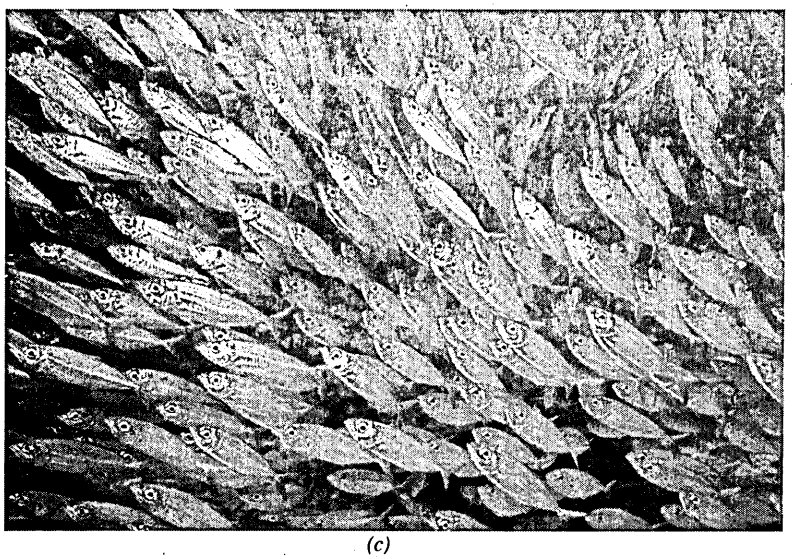
The works/studies in organised formation can be broadly separated into two distinct categories: *centralised* and *decentralised* formation. Centralised formation is where there exists an entity or more, acting as a supervisor or controller which can oversee the whole group and command each individual in the group accordingly. A well known biological example of the centralised organised formation is that of the sheepdog in which the system acts as a controller that controls and guards the movement of the sheep herd.

The second category is decentralised organised formations. In this category, there is no controller or supervisor to control the organisation and coordination of each individual. Each individual in the group reactively plans its next movement usually according to physical cues within its local neighbourhood. These physical cues can be anything in the environment; such as obstacles, other individuals in the neighbourhood range, or may be the intensity of the light. Examples in this category include line formation by ants, flocking of birds and schooling of fish.



(a)

(b)



(c)

Figure 2.4: Example of biological swarms. (a) Wildebeest herd grazing across Savannah Kenya (reproduced with permission from the Planet Earth Productions). (b) Wild parrots, wheeling in the sky, in Edgewater New Jersey, USA (reproduced with permission from Stephen C. Baldwin, brooklynparrots.com). (c) School of Silverside fish (reproduced with permission from R. Kent Wenger)

2.1.2.1 Centralised

In centralised organised formation techniques, a computational unit can oversee the whole group of agents and plans the action of the group individuals accordingly (De La Cruz & Carelli 2006)(Tanner & Kumar 2005). The action that should be taken by each agent is then transmitted to the agent via some kind of communication methods. There are not many works done in this category, after all it will defeat the purpose of swarm robotics which emphasises decentralised control.

De La Cruz and Carelli (2006) proposed a controller for positioning and tracking the desired agent formation. It operates in a centralised way and consists of two stages. At first a complete dynamic of a unicycle-like mobile agent and its linear parameterisation is modelled. Then the input-output feedback linearisation of the model is performed. On the second stage, the model of multi-agent systems is obtained by arranging all the feedback linearised agent models into a single equation. This multi-agent model is expressed in terms of formation states by applying a coordinate transformation. Finally the inverse dynamics technique is then applied to design a centralised formation control; which can be applied both to positioning and tracking the desired agent formations. They proved their method by using physical agents where, the agent formation errors for distance and angle errors are 0.05m and 0.03rad respectively after 17.5 seconds.

Tanner and Kumar (2005) introduced a navigation function through which a group of mobile agents can be coordinated such that they can form a particular formation, while moving in a group and avoiding collisions in the environment. In this approach, graph theory is used, where the properties associated with the interconnection graph are shown to affect the shape of the navigation function. The potential field produced by the function ensures that almost global asymptotic convergence of the agents to a particular oriented formation shape, while guaranteeing collision avoidance in the process. Although the proposed scheme is centralised, the potential function was constructed in a way that facilitates complete decentralisation.

Other centralised methods commonly used for mobile agents can also be used for multi-agent systems. Such methods include the many path planning algorithms. In path planning, there are many algorithms that have already been proposed. The efficiency of an algorithm can usually be evaluated in 4 different ways (Russell & Norvig 1995):

1. Completeness: Does the search can find a solution?
2. Optimality: Does the search can find the optimal solution?
3. Time complexity: How long is the time taken to complete the solution?
4. Space complexity: How large memory is needed to perform the search?

Path planning searches can be divided into two distinct categories, heuristics and stochastic searches. Heuristic search strategies use problem specific knowledge beyond the definition of the problem itself, thus can find solutions more quicker more efficient compare to stochastic search strategies. The are many well known heuristics search algorithms, which include the A* search (A-star search), Greedy best-first search, Memory-bounded heuristic search, Recursive best-first search (RBFS), and so on. However in this thesis, only the A* search will be introduced briefly, as it will be used as one of the basis for comparisons in one of the three contributions presented.

The A* search is one of the most widely used search algorithms. It is a best-first, graph search algorithm that calculates the least-cost path from a given initial node to another node. The nodes are evaluated by:

$$f(n) = g(n) + h(n) \tag{2.1}$$

where $g(n)$ is the cost to reach the goal, and $h(n)$ is the cost to get from the node to the goal. Since $g(n)$ gives the path cost from the start node to node n, and $h(n)$ is the estimated cost of the optimum (cheapest) path from node n to the goal, then the $f(n)$ is the estimated cost of the cheapest solution through node n. For that reason, the optimum local solution is the node with the lowest value of $g(n) + h(n)$; provided that the heuristic function $h(n)$ satisfies certain conditions.

It uses a distance-plus-cost heuristic function (usually denoted *f(x)*) to determine the order in which the search visits nodes in the tree. The distance-plus-cost heuristic is a sum of two functions: the path-cost function (usually denoted *g(x)*, which may or may not be a heuristic) and an admissible "heuristic estimate" of the distance to the goal (usually denoted *h(x)*). The path-cost function *g(x)* is the cost from the starting node to the current node.

2.1.2.2 Decentralised

Studies on organised pattern formations in a decentralised way are receiving increased attention in recent years. There are two distinct approaches to the coordination and organisation of multi-agent systems reported in the literatures; the first is the *behaviour based* approach, and the second is the *leader-following* approach.

Behaviour based approach

In the Behaviour based control approach, the systems often use relatively little internal variable state to model the macroscopic behaviour. The controllers consist of a selection of behaviours that maintain and/or achieve goals (Mataric 1999). For example, "collision-avoidance" will maintain the goal of preventing collisions and "homing" will achieve the goal of reaching some home destination. In more complex behaviours, some primitive behaviours of agents such as "collision-avoidance" and "goal seeking" are predefined, and the final formation control of agent is derived from a weighting of the relative importance of each behaviour. The advantage of the approach is that the group dynamics contain formation feedback by coupling the weightings of the actions taken. The disadvantages are that the group behaviour cannot explicitly defined, and the dynamics of the group are unpredictable making it hard to guarantee the stability of the whole systems (Takahashi *et al.* 2004),

Freeman *et al.* (2006) proposed an algorithm called the "distributed estimation algorithms" which allow agents in a communication network (or neighbourhood) to maintain the estimates of summary statistics describing the shape of the current swarm. In this study, each agent is able to control and organise its velocity and acceleration and also sense its own position, and exchange information with other agents within its neighbourhood. As a result, the agents form a

communication graph with changing topology as the agents move. Each agent implements an estimator that maintains an estimate of the current swarm formation statistics, based on its own sensed data and information received from neighbours, and a nonlinear motion control algorithm.

Nouyan *et al.* (2006) introduced the concept of chains with cyclic directional patterns; CDP-chains in short. CDP-chains are a method in robotic exploration of unknown environments. These chains are serve to explore the arena and establish a path between two points; food and home. Furthermore, the CDP-chains are also recruiting other agents to the food along formed path, and guide them to transport the food back to the home.

Desai (2002) proposed a graph-theoretical framework to control a team of agents moving in an arena with obstacles while maintaining a specific formation. The framework uses control graphs to define each agent behaviour or movement in the formation. The framework can also handle transitions between any two of the control graphs while avoiding obstacles. The complexity of computations for control graphs increases with the number of agents in the arena, however due to the facts that computations are decentralised, the framework described is scalable to a large group of agents.

Fierro and Das (2002) proposed another graph-based technique to tackle moving formations of a group of agents. They proposed a four-layer modular architecture for formation control namely, *group control*, *formation control*, *kinematic control* and *dynamic control.* Above all, group control layer is the highest layer which generates desired trajectories for the whole group to move. Formation control of a team of agents is built from three different networks namely *physical network*, *communication network* and *computational network.* It maintains the formation by using local communication and relative position information.

The kinematics control layer computes the required and angular velocities of agents. The dynamic control layer will finally deal with the task of realising the necessary speeds given by the kinematics control layer. The four-layer architecture represents an abstraction amongst tasks

required at different levels. For instance, agents with different dynamics such as mass, inertia and friction, can be used by changing the dynamics control layer on-the-fly, which in the end will promote reusability of the architecture. The reusability property makes the architecture very attractive for formal control applications, and will promote robustness to the systems.

Kaminka and Glick (2006) designed a multi-graph monitor framework for organised formation controllers that optimises the desired properties, for instance sensor usage for robustness. The framework consisted of two main strategies: cost optimal formation control graphs and dynamic switching of control graphs. Cost optimal formation control uses graph theoretics techniques that can be used to compute sensing policies that maintain a given organised formation, whilst dynamic switching of control graphs is a protocol allowing controllers to be switched on-line, to allow agents to adapt to sensory failures. Their results show that the use of dynamic protocol will allow formations of physical agents to move significantly faster and with greater precision whilst reducing the number of formation failures.

Yang *et al.* (2007) described an approach for controlling organised formations of multiple wheeled agents with parametric uncertainties and actuator saturations in the environment with obstacles. In this approach described, firstly, a collision-free trajectory is generated by introducing a non-convex optimisation problem. If the agents following the trajectory find that they are moving close to an obstacle, a new trajectory will then be generated by solving the optimisation problem under convex obstacle assumption. Secondly, to keep the agents tracking the reference trajectories or formations, a distributed moving horizon control scheme is used. Under this scheme, the whole optimisation problem is divided into several simple optimisation problems according to the number of cooperative agents, thus reducing complexity of the computation. Furthermore, close-loop properties inclusive of stability and robustness are guaranteed.

Pavone and Frazzoli (2007) developed a distributed control policy that allows agents to achieve different symmetric formations. The proposed scheme is inspired by the cyclic pursuit strategy, which is an attractive approach since it is decentralised and requires a minimum number of

communication links between agents to achieve organised formations. The proposed control policy generalises the notion of a classic cyclic pursuit algorithm by letting each of the agents pursue its leading neighbour along the line of sight rotated by a common offset angle. The key features with this method are stability of the systems and the possibility to achieve many different formations with the same simple control law.

Mastellone *et al.* (2007) introduced a control scheme that achieves dynamic formation control and collision avoidance for a group of nonholonomic agents. At first, for collision avoidance and tracking of a reference trajectory for a single agent, a feedback law using Lyapunov-type analysis needs to be derived. Secondly, by extending the derived result to the case of multiple nonholonomic agents, different classes of multi-agent problems involving an interacting group of nonholonomic agents such as formation control can be addressed. Finally, by combining the previous results, the problem of driving a group of agents according to a given trajectory while maintaining a specific formation can be addressed.

Cohen and Peleg (2006), studied and proposed a local spreading algorithm for mobile agents in 1-D and 2-D. In the study, oblivious or memory-less agents are used. The goal in this study is to spread N agents evenly within the perimeter of a given region. The algorithm for local spreading states that:

- first, each agent must first move to somewhere or some point so that it is at an equal distance with neighbours;
- secondly, the agents must move until there is no visible neighbour in the range.

For both 1-D and 2-D, at every time step it will calculate the average over all agents of the minimum distance to the nearest "object" (agent); (d_{av}) is defined as follow:

$$d_{av} = \frac{1}{N}\sum_{i} \min_{j \neq i} \{d_{i,j}\} \tag{2.2}$$

where the object considered in taking the minimum are all other agents and all points of the perimeter of the region. The next task is to move the agent to a point so that the agent will not perceive others within its vicinity.

In 1-D local spreading, agents are refer to according to their order on the line, and denote the position of each agent as i, $0 \le i \le N-1$, at time t, in the global coordinate systems by $R_i[t]$. The spread algorithm for agent in 1-D spreading is as follow:

- If no other agents are in sight, then do nothing.
- Otherwise, move to the point $\frac{R_{i+1}+R_{i-1}}{2}$.

In 2-D of local spreading the algorithm becomes a bit complicated. The algorithm is based on each agent i, dividing space into four quadrants Q_0 to Q_3, according the orientation as shown in the Figure 2.5. The spread algorithm for agents in 2-D spreading is given as follows:

- **For** $j = 0,\ldots, 3$ **do**:

  (a) $m_j \leftarrow$ coordinate of nearest agent or perimeter point in quadrant Q_j.

  (b) $d_j \leftarrow dist(i, m_j)$
- $q \leftarrow \text{argmin}_j \{d_j\}$; $d_{min} = \min_j \{d_j\}$: $d_{opp} = d_{3-q}$
- Move away from the current location by $\frac{d_{min}-d_{opp}}{d_{min}} m_q$.

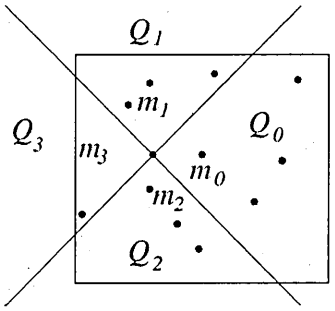


Figure 2.5: The four quadrants of agent's view. (Taken from Cohen and Peleg 2006)

Sun and Wang (2007) described a synchronous control approach to swarms of mobile agents in switching between different organised formations. At first, a position synchronisation error is defined as differential position error between every pair of two neighbouring agents; and it is derived according to the desired formation. Then a decentralised trajectory tracking controller is developed with feedback of position and synchronisation error. The developed trajectory

tracking controller is proven and guarantees that asymptotic convergence towards zero of both position and synchronisation errors. From their simulation, the results demonstrate the effectiveness of the proposed synchronous control design for the formation control.

Antonelli *et al.* (2005) proposed a technique for formation control of multi-agent systems. The proposed technique which they named Null-Space-Based Behavioural Control, is a behaviour based technique which aimed at coordinating a group of mobile agents while performing different missions. The missions are firstly decomposed into several elementary tasks and, for each of the task; a motion reference command for each agent is elaborated referring to a kinematic approach. The technique is then combined with the output required by each task in order to obtain the final motion command for each agent. In properly handling multiple, eventually conflicting tasks, it uses a hierarchy-based approach that uses the null-space projection. From their simulation results, they showed that null-space-based behaviour control offers the advantage to ensure the achievement of the output of the higher-priority task without being affected by the output of lower-priority task (Antonelli *et al.* 2006, 2007). However, due to its analytical nature, the proposed technique needs the definition of a suitable task function that admits computation of a proper Jacobian, which may be obvious for some tasks.

Nguyen and Do (2006) proposed a constructive method to design cooperative controllers based on local potential functions. The cooperative controllers are the controllers that force a group of mobile agents to achieve organised formation while avoiding collisions with other agents. Firstly, simple point-mass agents are considered to clarify the design philosophy. The technique is then extended to non-holonomic agents, and finally local potential functions are constructed to design gradient based cooperative controllers. This cooperative controller is designed to achieve almost global asymptotic convergence of a group of mobile agents to a particular formation in term of both shape and orientation, with a guaranteed of no collision between agents.

Avrutin *et al.* (2007) introduced the concept of connecting objects via random growing trees (RGT) in swarms of agents. In the working arena, there are at least two objects in addition to

swarm agents. One of the object is labelled the *base* and the others is the *target* objects. The goal is to "bridge" or connect the base and target objects by using the swarm agents. At the beginning, agents are uniformly distributed over the arena. The bridging task consists of finding every object, encircling the objects, and connecting them (goal-target) by lines of agents that should be as short as possible. The RGT approach can be split into three main phases:

- Exploration: Agents explore the arena, looking for objects (goal or target) and encircle the object.
- Formation of trees: Position one chain at the circle around the object and begin to build a tree out of chains using open end of this first chain as the root.
- Reduction of the tree: After all objects have connected to every other object, unneeded chains have to leave the tree. By doing this, the remaining lines of agents should be reduced to the minimal necessary number of agents needed to connect the objects.

As an addition, the formation of a first tree may already begun at one object whilst another one (object) has not been found yet.

Leader-follower approach

In the leader-following approach, some agents will act as leaders while others as followers. The leader agents track predefined reference trajectories, and the followers track transformed versions of the states of their nearest neighbours according to some given methods or schemes. The advantage of this approach is that it is easy to control multiple agents in a desired formation using only two different controllers and it is suitable for describing the formation of robots (Takahashi *et al.* 2004). Furthermore it is easy to understand and implement; even if the leader is perturbed by disturbances, the formation can still be maintained (Nguyen & Do 2006). The disadvantages in this approach includes the difficulty to consider the ability gap of an agent (Takahashi *et al.* 2004), there is no explicit feedback to the formations and if the follower is perturbed, the formation cannot be maintained (Nguyen & Do 2006).

Das *et al.* (2002) studied and proposed a vision-based framework for the development of non-holonomic multi-agent systems by composing *simple sensing*, *estimation*, *control*, and

coordination blocks in a bottom-up approach. The framework allows the designers to build complex multi-agent systems especially for leader-following structure algorithms from simple controllers and estimators. The main key features of the approach are a suite of control and estimation algorithms, and a model for switching that allows a group of agents to maintain a prescribed formation. The switching model will also allow the agent to change its formation in the presence of obstacles.

Takahashi *et al.* (2004) proposed a controller which is based on the ability of agents to use the leader-following strategy organised formation. The proposed scheme consists of three steps. First, a performance index for each agent, such as maximum acceleration, maximum velocity and maximum torque of a motor, is quantified. Secondly, based on the performance index or based on the ability of the agent, a new controller is then proposed. Finally, for collision avoidance, a compliance controller using virtual repulsion was proposed. Takahashi also showed that by using the proposed scheme, agents in leader-following formation can keep the formation even if the leader is changed.

Javaid *et al.* (2004) proposed a distributed control algorithm where organised formation of agents can be grown dynamically by using local sensing and minimal communication. In this algorithm, the controller on each agent consists of four different behaviours namely, leader, wanderer, member and candidate. The leader agent is predefined whenever the system is initialised. The leader will maintain the formation and heading, where only the leader knows the goal information. For the wanderer, the agent is programmed with the behaviour where its task is to look for the formation. The member agent, is programmed with the behaviour when the agent is part of the formation; when this behaviour is active, the agent will follow its immediate leader and maintain a fixed distance to it. Candidate, is the behaviour when an agent finds the formation and communicates with the other group members that it is going to attach with in the formation. The communication between a candidate and its neighbour or immediate leader is used for information exchange that consists of the type or shape of the formation, the number of agents currently in the formation, maximum number of agents allowed for the

formation, and any other formation parameters (radius of the circle for instance). Once all the necessary information is obtained, the agent will calculate its pan angle to keep in view of its neighbour and the distance from it. The formation of agents grows from a single agent to a maximum possible number of agents while in motion. The first agent, *i.e.* the leader, are predefined as mentioned previously. Others will try to join or make formation with the leader. Whilst in the formation, agents will try to maintain a regular polygon shape and hence make a virtual circle if the number of agents in the formation is adequate. The control algorithm is minimal but lacks the ability to maintain formation in the existence of obstacles.

Gustavi and Hu (2005) proposed control algorithms for multi-agent systems with limited sensor information. The proposed control algorithms are only based on agents with local information and without global knowledge. In the first algorithm, vertical tracking is designed such that the follower agent follows the leader agent's trajectory while maintaining the distance towards the leader. In the second algorithm, horizontal tracking is designed to make the follower agent move side by side with the leader agent at some predefined distance while maintaining same orientation as the leader agent. The third control algorithm is combination of vertical and horizontal tracking; in this algorithm Gustavi and Hu showed that by combining the first two algorithms, more complex multi-agent organised formation can be formed. However, the stability of the systems which can be affected by switching between the first two different algorithms yet remain to be shown.

Li *et al.* (2006) focused on the leader-follower type of organised formation control algorithm of multiple differential-driven wheeled mobile agents. The proposed control strategy is derived from the dynamics of the agent directly. The control strategy takes the acceleration ability of the agent into account and uses only its local sensing data and small data communication to achieve organised formation control. From the experiments shown, whenever the leader agent tracks different trajectories, the follower agents can always adjust itself to form the desired formation as quickly as possible, and it will maintain the formation stably over time. From the extensive experiment done by Li *et al.*, it shown that the method is quite effective for formation

establishing and stable for the formation within their abilities.

Chen *et al.* (2007) proposed a decentralised formation control system based on dynamic regulations and scheduling scheme. From the simulations of a typical leader-following triangle formation (or V-formation), the trajectory of the group can be calculated in advance or can be planned in real time by the leader. The leader approaches its desired goal in an arc-type trajectory, therefore the real trajectory, being piecewise-smooth, can be obtained. Furthermore, the followers adjust and maintain the formation shape with the piecewise-smooth arc trajectory as well. While maintaining the formation shape, the control regulation switches internally between OTR (Offset regulation) and SDR (Spacing distance regulation) depending on the dynamic formation framework of the formation. Thus it promotes adaptability which is very attractive in the multi-agent systems.

Sorensen and Ren (2007) introduced a unified formation control scheme using the leader-follower approach with consensus-based formation control. In this scheme, an agent requires only local neighbour-to-neighbour information exchange. In addition, an extended consensus algorithm is applied to estimate the time varying group trajectory information in a distributed manner. A consensus-based distributed formation control strategy is then applied to each agent based on the estimated group trajectory information. Sorensen and Ren also studied the effect of the multiple group leader and found that by increasing the number of group leaders within the formation, agent estimate of the formation state is improved and the system is robust against single point failure.

Fredslund and Mataric (2002) used a neighbour referenced method, where each of the follower agent keeps a single "friend" at a desired angle ϕ, using some appropriate sensor. The angle ϕ is predetermined in a particular type of geometric pattern. When agents flock, a leader will navigate a path while the follower agents maintain the angle and distance to their neighbours.

Parker *et al.* (2004) proposed a tightly-coupled cooperation in heterogeneous agents performed by two different types of agents, namely *leader* and *simple* agents. In this strategy, *leader*

agents which have rich sensing capabilities assist *simple* agents with limited capabilities for navigation and obstacle avoidance. But such a strategy makes the leader more costly and the team becomes less robust to the failure of the leader.

2.2 Definitions

2.2.1 Intelligence

Kennedy *et al.* (2001), in their book, stated that:

> *"Intelligence is a word usually used to describe the mental abilities of humans, though it can be applied to other organisms and even to inanimate objects like computers and computer programs. There is very little agreement amongst psychologists and amongst computer scientists about what this word means, and almost no agreement between these two groups".*

Fogel (1995) claims that a good definition of intelligence should apply both to humans and machines equally well, and believes that the concept should apply to evolution as well as to behaviours perceptible on the human time scale. Fogel concluded in his paper that intelligence, whether in an animate or inanimate context, can be defined as the "ability of a system to adapt its behaviour to meet its goal in a range of environment".

The discussion of intelligence in computer science is often intertwined with the Turing Test (Turing 1950). Turing gave intelligence a simple definition:

> *"intelligence is fundamentally the ability to solve problems, particularly unusual or new problems".*

The Turing test itself sounds simple enough where a subject is placed in a room with a keyboard and a monitor, while in another room there is a computer and a person. The subject will then type questions into the keyboard and receives a reply from the other side of the room. A summary of the test is: if the subject is unable to tell if the computer's responses were

generated by human or the machine, then the machine is considered intelligent (Kennedy *et al.* 2001). This test has been subject to different kinds of criticism and has been at the heart of many discussions in AI, philosophy and cognitive science communities for the past 50 years.

Beni (2005) commented on an example of a manufacturing plant, in which a manufacturing machine which produces a mechanical piece for a car in an ordered manner but in a predictable way is not considered as intelligent. Likewise, even the rolling of dice where the outcome is unpredictable and does not produce order is not considered as intelligent. He then concluded that, the main characteristics of intelligent behaviour is the production of something ordered and the outcome should not be predictable, *i.e.* emergent intelligent behaviour.

Dorigo and Schnepf (1993) believe that intelligent behaviour cannot be created in artificial systems without the ability to interact with a dynamically changing unstructured environment. They added that cognition of the robot emerges only when autonomous systems try to impose structure on the perceived environment in order to survive. These structures in turn provide the ground work for more intelligent behaviour such as; the skills to learn, the emergence of goal-directed behaviour and the development of problem-solving methodologies. These basic cognitive skills have been developed as part of the evolutionary process, and unlikely to have been present in biological systems from the beginning of the life.

The research in this thesis address the problem of pattern formation of robot swarms or swarm systems. Earliest work on Swarms comes from the research on social insects. Biologists have been inspired by cooperative behaviours of insects like ants and bees, which led to intense research on their behaviours. The central to the swarm systems are decentralisation and self-organisation which promote the emergence behaviour.

To date, there many definitions of intelligence have been defined by several groups. The work in this thesis is related to robot swarms or swarm systems. As this thesis is related to swarm systems, this thesis defines brief definition of intelligence as:

"the ability to solve problems in unpredicted way".

2.2.2 Swarm Intelligence

Insects that live in colonies such as ants, bees termites and wasps have been living on earth for millions of years, building nests, organising production and foraging for foods. It has been known that they care about order and cleanliness. They have a simple communication mechanism and warning system, maintain an army and divide labour. In addition, they are very flexible and can adapt well in the changing environment. This flexibility makes the colonies robust (Bonabeau *et al.* 1999).

In 1989 when Beni and Wang were investigating the properties of simulated, self-organising agents in the framework of cellular robotic systems, they introduced the concept of Swarm Intelligence (SI). SI is an artificial intelligence technique based around the study of collective behaviour in decentralised, self-organised systems. It is composed of unintelligent individuals, but the group demonstrates complex behaviours (Bonabeau *et al.* 1999). Such systems are typically made up of a population of simple individuals which interact locally with one another and with their environment which lead to the emergence of global behaviour.

The main feature of self-organisation is that a system's organisation or movement does not explicitly depend on external control factors. In other words, the organisation emerges solely due to the local interactions between individuals and their environment (Camazine *et al.* 2001). The organisation can evolve dynamically either in time or space and can maintain some kind of stable form or can show transient phenomena. An example of such a system is that of a colony of ants sorting eggs without any particular ant knowing the sorting algorithm itself (Bonabeau *et al.* 1999).

Like the word *intelligence*, the definition of emergence (or *emergent behaviour*) has attracted the attention of some researchers'. Taylor (1990) asserts that the emergent properties are collections of units at a lower level of organisation and, through their interaction, often give rise to properties that are not the mere superposition of their individual contributions.

Steels (1991) describes "emergent functionality" as a function that is not achieved directly by a component or hierarchical system of components, but indirectly by the interactions of more primitive components amongst themselves and with the world. Mataric (1993) defines emergent behaviour for swarm intelligence as follows:

> *"emergent behaviours is apparent by global states which are not explicitly programmed in, but it results from local interactions amongst individuals. It is considered interesting based on some metric established by the observer".*

Despite several differences in the definition of emergence, one common theme connects all these definitions in the AI (Artificial Intelligence) community, *i.e.* emergent behaviour occurs as a result of local interactions amongst individuals and between individuals and their environment.

Many social insect societies exhibit interesting complex behaviours in organising themselves to perform specific activities such as foraging and nest building. Cooperation amongst individuals arises through an indirect communication mechanism, called stigmergy and by interacting through their environment (Holland & Melhuish 1999).

Stigmergy is a word coined by the biologist Grassé in the 1950s. The word itself was used to explain the task coordination and regulation in the context of nest building by termites. In termites nest building activities, Grassé showed that the activities do not depend on the individual workers themselves but mainly are achieved by the current nest structure. The current local nest configuration is of course was configured by the previous termite activity in which will trigger the current activity or configuration of the local area of the nest. The configuration will then again stimulate the response of a same termite or a different one in the colony.

Although there is normally no centralised control structure dictating how individual agents should behave, local interactions between such agents often lead to the emergence of collective

behaviour. Such systems can be found in nature and include ant colonies, bird flocking, fish schooling, animal herding, bacteria moulding *etc.*

Interactions between individual insects in social insect colonies have been well documented. Some examples of such behaviours are bee dancing and ants' pheromone trail-laying during food foraging. These simple communication systems between insects lead to the collective intelligence of social insect colonies.

Take ants as an example. First, an ant takes a bite from a food source and then wanders off. After a short while, lots of ants will begin to queue in neat lines to and fro, following what seems like the shortest route between the food and the nest. It seems like ants have some kind of higher intelligence, and yet ants only have several hundred neurons to help them consider what to do next. In fact, ants do not plan, they just react to their environment (Bonabeau *et al.* 1999).

Many ant species have trail-laying and trail-following behaviour when foraging. When an ant stumbles across a piece of food, it does not remember where it is, it just deposits a chemical trail using pheromones as it moves from food source to its nest. To find food, others will follow these trails (Franks *et al.* 1991). At first, ants choose between a long and short path at random, but because more ants travel the shorter path in a given time, the pheromone trail reinforces the pheromone signal. This will become the favoured path. This method of using the world as a memory bank is the aforementioned stigmergy. The process where an ant is influenced towards a food source by another ant or by a chemical trail is called recruitment. Recruitment based solely on pheromone trails is called mass-recruitment (Franks *et al.* 1991).

As with ants, the self-organisation in honeybees is also based on relatively simple rules of individual insect behaviour. The rules specify that the interactions amongst the system's constituent units are executed on the basis of purely local information, without reference to the global pattern. This is an emergent property of the system rather than a property imposed upon the system by an external ordering influence.

Self-organisation of honeybees is based on simple rules of individual insect behaviour (Von-Frisch 1968). When a bee finds a nectar source, it then goes back to the hive and relinquishes its nectar to a hive bee. Each hive has a so-called dance floor area in which the bees that discover nectar sources dance in a way to recruit and convince their hive-mates to follow them. After the nectar is exhausted, the bee can either:

- abandon the food source and become an uncommitted follower again, or
- continue to forage at the food source without recruiting hive-mates, or
- dance and thus recruit the hive-mates before it returns to the nectar source.

The dance is a communication mechanism used for recruitment in honeybees, in which the information about distance, location and quality of a nectar source is also transmitted.

Within the dance area, the bee dancers "advertise" different food areas. The mechanism by which a bee decides to follow a specific dancer are yet to be well understood, but it is always considered that the recruitment amongst bees is always a function of the quality of the food source (Camazine & Sneyd 1991). From the experiments conducted by Camazine and Sneyd (1991), they confirm that not all bees start foraging simultaneously, but begin foraging at a rate proportional to the difference between the eventual total and the number presently foraging.

By writing programs that model the natural behaviours of swarming animals or insects, programmers in the field of Computer Science can solve many complex problems. In swarm applications, the agents working on the problem usually have no knowledge that a problem even exists, they are in fact just continuing with their "natural" behaviour, and it is that behaviour that helps solve the problem.

In recent times, many researchers have shown an increasing interest in building multi-robot systems or, on a much larger scale, robot swarms. Unlike other studies on multi-robot systems in general, swarm robotics emphasises self-organisation and emergent behaviour in a large number of agents whilst promoting scalability, flexibility and robustness of the system by only using limited local capabilities. This also requires the use of relatively simple robots, equipped with limited communication mechanisms, localised sensing capabilities and the exploration of

decentralised control strategies.

2.3 Swarm Robotics

Swarm robotics is a new approach for the coordination of multi-robot systems which consist of large numbers of relatively simple physical robots. The goal of this approach is to study how relatively simple physical embodied agents can be constructed to collectively accomplish tasks that are beyond the capabilities of a single agent. Sahin (2005) gave a formal definition of swarm robotics as follows:

> *"Swarm robotics is the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions amongst agents and between the agents and the environment".*

Cao *et al.* (1997) suggest that the earliest study on swarm robotics was started in the early 1970s, although there was limited interest in the area. At the time, coordination and interaction of multiple agents were being focused in the field of distributed artificial intelligence, DAI for short (Cao *et al.* 1997). However the investigations were limited to the problems involving software agents. This tendency remained until the late 1980s, when roboticists began to explore cooperative robotic systems (Arai *et al.* 2002).

One of the earliest studies in the cooperative robotics field was related to cellular robotics systems. Cellular robotic systems such as CEBOT (CEllular roBOT) was initially studied by Fukuda *et al.* (1989). The CEBOT system is a robotics system which consists of several homogenous agents in the system. The agents, which they refer to as "cells" can make connections and separations between them, which in turn will reconfigure the structure of their systems. Moreover, the system is able to reconfigure itself into an optimal structure depending on purpose and environment.

Unlike other studies on distributed robotic systems, in general, swarm robotics emphasises self-organisation and emergence in large numbers of robots and promotes scalability and robustness by using only local communication. These emphases promote the use of relatively simple robots, equipped with scalable communication mechanisms, localised sensing abilities and the exploration of decentralised control strategies.

Moshtagh *et al.* (2006) developed vision-based control laws for flocking on non-holonomic swarm agents. In this approach, agents are fixed with a camera that has a fish-eye lens capable of seeing the entire surrounding of the agent (with a field of view of 360°) for visual measurements of velocity alignment. The controller will need the values of bearing, optical flow and time-to-collision, all of which can be measured from images taken from the camera. From their simulation results, there are visible differences on the convergence rates between the noise free and noisy environments. For the noisy environment, a Gaussian random noise was added to the measurements of bearing.

Esposito and Dunbar (2006) controlled the coordination of swarm agents towards multiple sub-goals while maintaining some range of wireless connectivity with a line-of-sight constraint between agents in the presence of obstacles. To solve the problem, they proposed a method for composing multiple potential functions, which indicate a set of possible input directions into a single feasible movement direction from the condition that the state vector of the agent approaches the minima of the potential function.

The first potential function is the Navigation Function. The Navigation Function is the basis for ensuring the goal completion portion of the problem ($q \rightarrow q^f$) is at least achieved. Generally, Navigation Functions are artificial potential fields that simultaneously provide obstacle avoidance, and almost always, convergence to a goal configuration. The Navigation Function for agent i is define as follow:

$$\phi_i^{goal}(q_i) = \frac{d^2(q_i, q_i^f)}{[d^k(q_i, q_i^f) + \Pi_{j=0}^{M} d(q_i, O_j)]^{1/k}} \tag{2.3}$$

Where O_j is obstacle j, O_0 is the boundary of the workspace, $d(q_i, q_j)$ denotes distance of agent

q_i and agent q_j, and k is a parameter. The parameter k must be selected to be high enough so that all local minima, except at q_i^f disappear.

The second potential field is the range. The range between agent q_i and q_j is define as follows:

$$\phi_{ij}^{range}(q_i, q_j) = \begin{cases} 0 & \text{if } d(q_i, q_j) < \rho_{max} \\ d^2(q_i, q_j) - \rho_{max}^2 & \text{otherwise} \end{cases} \tag{2.4}$$

Where ρ_{max} is the maximum range of agent q_i and agent q_j. The potential only possesses minima at configurations where the constraints are satisfied, but is not strictly a navigation function.

The third potential is the Line-of-Sight (L.O.S.). If two agents q_i and q_j are in danger of loosing sight of each other, it means that one of them (*e.g.* agent q_i) is occluded from the other's (*e.g.* agent q_j) view by an obstacle. The line connecting the two agents at the last time when L.O.S. was satisfied is referred to as the *occlusion line*, *OL*. The line of sight constraint is enforced by a following potential:

$$\phi_{ij}^{los}(q_i, q_j) = \begin{cases} 0 & \text{if } L.O.S. \\ d^2(q_i, OL) & \text{otherwise} \end{cases} \tag{2.5}$$

Where $d^2(q_i, OL)$ denotes the distance from agent q_i to the occlusion line. The potential only possesses minima at configurations where the line-of-sight constraint is satisfied, but does not serve as a proper Navigation Function.

In the technique described above, all the agents must pass on the same side of an obstacle for the agents to remain connected. In order for all the agents to remain connected, the swarm must either have a leader of some sort, or some on-line method for achieving consensus on which path to take. Another consequence of this technique is that due to the existence of saddle points, the swarm is occasionally unconnected.

With the recent technological advances, the development of swarm robotics is becoming more and more feasible. There are already a number of on-going and completed projects that aim to

develop and/or control large numbers of physically embodied agents. Such projects are discussed herewith.

2.3.1 The Autonomous Nano-Technology Swarm (ANTS)

The Autonomous Nano-Technology Swarm (ANTS) is a project funded by NASA (National Aeronautics and Space Administration, USA) (Curtis *et al.* 2000). In this project, the mission is to develop a swarm of autonomous satellite agents that will search the asteroid belt for asteroids with specific characteristics. There will be around 1000 agents involved. Each agent will have a high degree of total or near total autonomy. The social structure of agents is based on hierarchy by using heuristic approaches. Agents also have the ability to modify their operation autonomously. This is crucial for agents to reflect the changing nature of the mission, the distance, and the low bandwidth communication back to earth.

In the mission, agents are divided into three categories which is based on the agent's ability:

- Leaders: the leader will have rules and goals for the mission; the leader will also coordinate the work effort of the worker agents.
- Workers: the workers will perform tasks given by the leader and follows the rules for the mission.
- Messengers: messengers will relay and coordinate communications and information between leaders, workers and the Earth.

Leader agents are equipped with models of the types of science they want to perform. Parts of the model include the ability to communicate with messenger agents that then relay the information to the worker agents. Teams of agents will carry out the work together to form models of asteroids as well as form virtual instruments.

2.3.2 The Swarm-bots project

The Swarm-bots project funded by the European Community (Dorigo *et al.* 2004b). The project lasted for 42 months and completed in March 2005. The main objective of the project is to seek new approaches to the design and implementation of self-organising and self-assembling

mobile agents. The agents are composed of a number of simpler, insect-like agents (called s-bots), which are capable of self-assembling and self-organising to adapt to its environment.

There are three sets of objectives in the Swarm-bots project:

- dynamic shape formations,
- navigation on rough terrain, and
- scaling up.

In the dynamic shape formation, the s-bots are able to self-assemble into a number of different planar and 3-D geometric configurations, for instance like those formation found in ant colonies and in patterns of differential adhesion by developing cells.

In navigation on rough terrain, the s-bots will be able to move across the terrain arena guided by sensory information gathered by the individual s-bots. There are three sub-objectives that have been defined:

- S-bots should be able to maintain the original or current shape configuration while following light.
- S-bots should be able to reconfigure automatically while following light through narrow passages and tunnels.
- S-bots should be able to reconfigure their shape to pass over a hole or through a steep concave region that could not be passed by a single s-bot.
- S-bots should be able to move from point A to B on rough terrain on a shortest possible trajectory.

Finally, the objective in the scaling up is to study the impact to the swarm-bots robotic systems when the user increases the number of the s-bots in both categories as described previously.

2.3.3 The Pheromones robotics project

The Pheromones robotics project funded by DARPA (Defence Advanced Research Projects Agency, USA). This project seeks approaches to the design and implementation of self-

organising and decentralised control of the robots (Payton *et al.* 2004).

In this project, Payton *et al.* (2004) developed a realistic model of the pheromone based communication of ants by using eight directional infrared transmitter-receiver pairs attached to the top of the agents. The pheromones are assumed to be transferred between the agents as 10-bit messages by using the infrared transmitter-receiver. Each agent then retransmits the message it gets by reducing the intensity of the pheromone and decrements the hop count in the opposite direction. This method is used mainly to generate the path between two points in an unknown area by a swarm of agents.

2.3.4 The GUARDIANS project

The GUARDIANS (Group of Unmanned Assistant Robots Deployed In Aggregative Navigation supported by Scent detection) project is a three year programme funded by the European Community and started in January 2007. The objective of the project is to develop a swarm of autonomous robots that consists of several robots that will navigate through an industrial warehouse in smoke or on fire. Amongst the possible tasks that agents should be able to perform are: searching the warehouse to explore the environments and gather information for map building, and supporting the firemen to move around in the environment while avoiding obstacles (Penders *et al.* 2007).

2.4 Robot Architecture

Many works in the swarm robotics field are inspired by the behaviour based control architecture (Brooks 1986)(Arkin 1998)(Balch & Arkin 1998). One of the pioneers in the behaviour based field was Braitenberg (1984). He describes a series of thought experiments in which "vehicles" with simple internal structure, where sensors are directly coupled to the motors, behave in unexpectedly complex ways. He developed simple control architectures and created a wide range of vehicles producing sophisticated emergent behaviour, which he then labelled with terms such as aggression, cowardice, fear, foresight, love and even optimism. Braitenberg gives this as evidence for the "law of uphill analysis and downhill invention".

However, these systems were inflexible and were not reprogrammable. A possible example of Braitenberg vehicle is illustrated as in Figure 2.6.

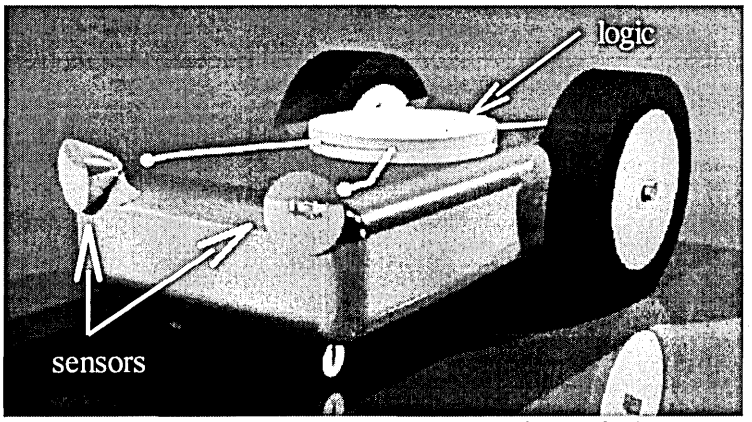


Figure 2.6: Artist impressions of Braitenberg vehicle.

In 1986 Brooks introduced the subsumption architecture (see Figure 2.7), where each task-achieving behaviours are represented in separate layers. Individual layers work on individual goals concurrently and asynchronously. At the lowest level, each behaviour is represented using a finite state machine model, and higher levels are allowed to subsume the activity of the lower ones but not the other way around.

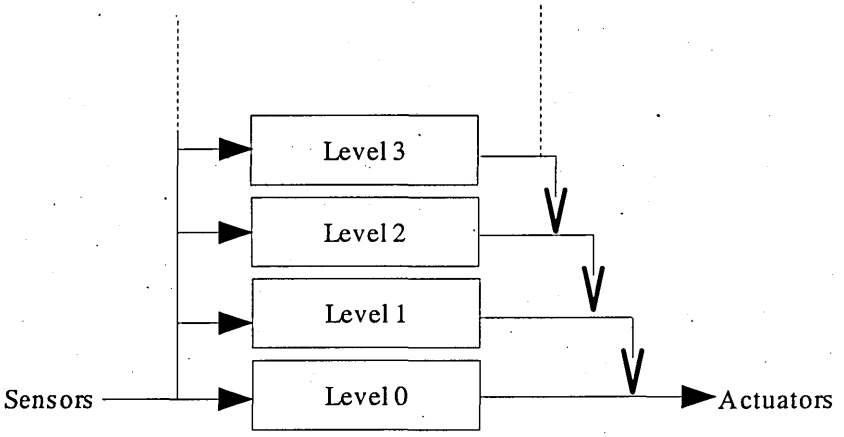


Figure 2.7: The Brook's subsumption architecture.

Brooks (1991b) also stated that, in order for an autonomous mobile agent to be considered intelligent, the agent must be robust and extensible, and have multiple goals and sensors. In this architecture, although the robot has multiple goals, not all sensors reading are adopted. Only the

ones with perception processing identified as extremely reliable are to be used. An advantage of the system is that it is inherently modular from a software design perspective, and it enables the robotics system designer to expand the agent competency by adding new behaviours without redesigning the old ones. Brooks (1991a) later suggested that researchers should focus on highly simplified intelligent systems rather than having an unrealistic goal of replicating the level of human intelligence.

Although the subsumption architecture has many good characteristics, in many cases, multiple and possibly conflicting goals cannot be achieved (Rosenblatt & Payton 1989)(Barnes *et al.* 1997). As the name *subsumption* implies, conflicting goals are often resolved by having one behaviour's commands completely override the other's. Even though it may be highly desirable for the systems to act simultaneously, to accommodate the needs of both behaviours there is no way to arrive at a balanced solution.

Another early behaviour based mobile agent control architecture is known as the motor schema-based architecture (Arkin 1989). Schemas are a methodology used to describe the interaction between perception and action. It can be adapted to yield a mobile agent system that is highly sensitive to its currently perceived world. Motor schemas operate in a concurrent, independent and communicating manner, which can produce paths that reflect the uncertainty in the detection of the objects and yet, can cope with conflicting data arising from diverse sensor modalities and strategies.

Up to the mid 1990s, many researchers were of the opinion that behavioural agents were incapable of achieving more complex tasks than simple can collecting, box pushing, herding or moving in formation. Problems such as behaviour conflict resolution, behaviour adaptation and behaviour scheduling had been identified as the main issues for multiple mobile agents to co-operatively perform a complex task.

Several approaches have been developed to address these issues. One of the approaches is known as the Behaviour Synthesis Architecture (BSA) (Barnes *et al.* 1997). BSA has four

different behaviour levels called: *self*, *environment*, *species* and *task*. Sensory stimuli (in BSA) provide the appropriate internal and external state information needed for the various behaviour levels and from each relevant level, appropriate motion responses are generated that relate to the desired actuation. In any behaviour level, there are a number of *behaviour patterns* (**bp**'s), where it defines what a robot's motion response should be for a given sensor input, and also provides a measure as to how the relative importance of the response varies with respect to the same sensor input.

2.5 Artificial Life

Conventional artificial systems are usually designed strictly in a *top-down* manner. In this approach, the systems are designed to function precisely and effectively for special purposes and specifically under closed domain. Thus, these systems fail to respond appropriately to unexpected situations. On the other hand, for natural systems as well as their entire behaviours emerge through *bottom-up* processes. These natural systems do adapt themselves quite well in their environment that exhibit dynamic and unpredictable characteristics. Moreover, they also can cope with a variety of difficulties.

2.5.1 Inspirations from Natural Systems

In the field of Artificial Life (Alife), where models are based on the natural systems, generally a *bottom-up* approach is used (Bedau 2003). The work in Alife can be loosely divided into three categories: *soft*, *hard* and *wet,* where *soft* is software based, *hard* is hardware based, and *wet* is biochemistry based. The essential features of Alife models were given by Langton (1988) as follows:

- they consist of populations of simple agents;
- there is no single agent that directs any of the other agents;
- each agent's specifications (program) details the way in which it reacts to local situations within its environment, including encounters with other agents;
- there are no rules in the system which dictate global behaviour, therefore

- any behaviour at levels higher than the individual agents must be emergent.

One of the earliest models of Alife was modelled by Newmann (1966). Newmann designed the Alife model when he created his well known self-reproducing, computational-universal cellular automata. Newmann was first intended to built physical self-rebuilding robots and the design was known as the kinematic model. As the work progressed, he realised that the huge cost of providing the robot with parts of which to build its replicants. Because of this huge cost, he then developed his design around mathematical abstraction, thus creating the first implementation of cellular automata.

Another early example of Alife was created by Walter in 1950. He constructed two mobile autonomous robots using valves and light sensors, named Elmer and Elsie. Elmer and Elsie were programmed to search for a set level of light intensity. Upon seeing light, they will move towards the light; if the intensity of the light is too strong, they will move away from it. Whenever the power is running low, they will return to the hutch to recharge their power. Next, Walter fixed a light on both of them. At first, they moved towards each other and engaged in the fascinating dance which he described in his book, "The Living Brain" (Walter 1963). However when the light in the hutch is switched on, Elmer and Elsie will stop "dancing", ignore each other and head towards the hutch.

There are many works and developments in Alife which are relevant to this thesis, one of them is the artificial fishes in a 3-D virtual physical world which was developed by Terzopoulos *et al.* (1994). They emulate the individual fish's locomotion, behaviour and appearance as an autonomous agent situated in its simulated physical domain. Moreover, the fish can learn how to control their internal muscles to move hydrodynamically. They also emulate the complex group behaviours in a certain physical domain.

Figure 2.8 shows the functional overview of the artificial fish. The body of the fish houses a brain or mind with motor, perception and learning centre. The motor system consists of actuators and a set of motor controllers, which drive the dynamic model of the fish. They

(Terzopoulos *et al.* 1994) built the motor controllers unit by gleaning information from the fish biomechanics literatures, thus their fishes can swim realistically. In the perception centre, the perception of the fish relies on a set of on-board virtual sensors to provide sensory information about the dynamic environment. In the brain part of the perception centre there is a perceptual attention mechanism which allow the fish to train its sensors at the world in a task-specific way, which will then filter out sensory information that is unnecessary to its current behavioural needs.

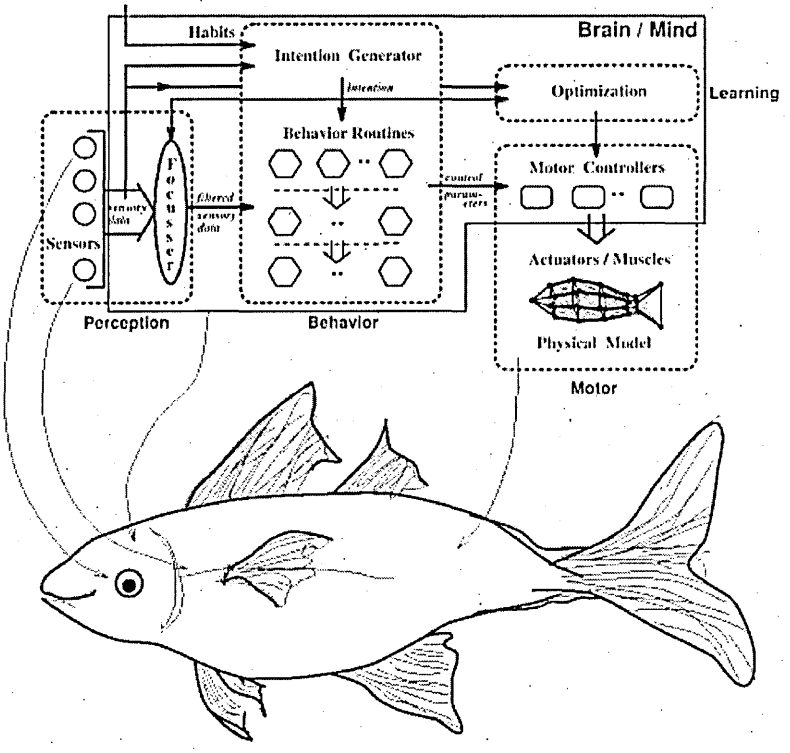


Figure 2.8: Control and information flow in artificial fish. (Taken from Terzopoulos et al. *1994)*

The behaviour centre of the artificial fish acts as a medium between its perception system and its motor system. The intention generator in the behaviour system is the fish's cognitive faculty which harnesses the dynamics of the perception-action cycle. Finally, the learning centre enables the artificial fish to learn how to locomote through practice and sensory information. The learning centre also enables the fishes to train themselves to accomplish higher levels of sensorimotor tasks, *e.g.* manoeuvring to reach a visible target.

Schmickl and Crailsheim (2007a) developed an algorithm for swarm robotics navigation based on a technique of signal propagation seen in slime mould as in Figure 2.9. In this technique, agents will wander and search randomly in an arena. When a target is found by an agent, the agent will then send out a signal using LED flashes notifying other agents about the location of the target. The signal is detected by the others, who will then forward or re-transmit the signal using the same method. The process will go on and on, resulting in a wave-like signal propagation such as that exhibited by slime mould.

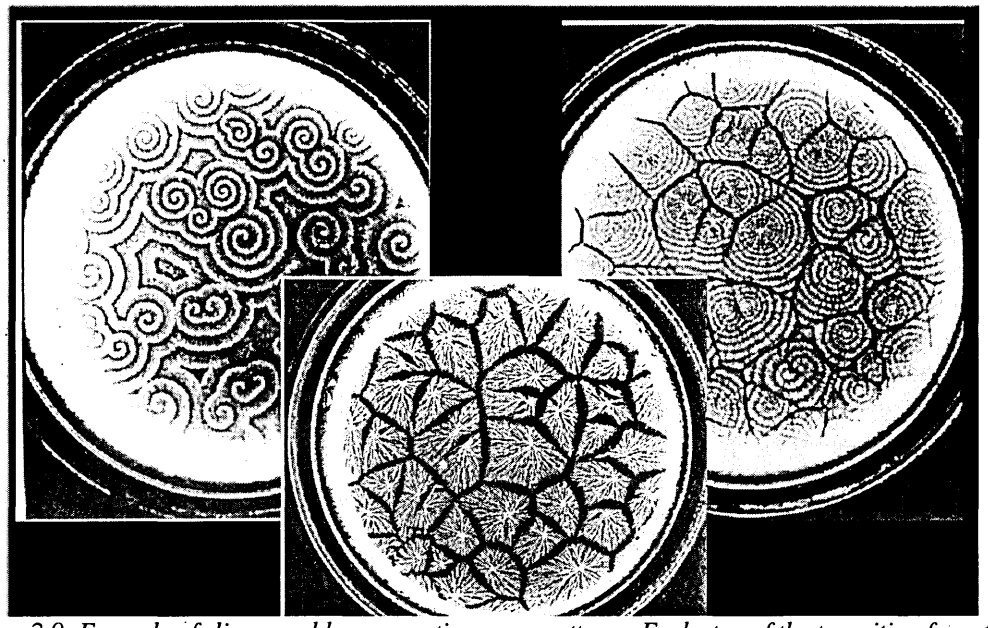


Figure 2.9: Example of slime moulds aggregation wave patterns. Each step of the transition from top left to top right and then to the centre takes about 30 minutes. Images courtesy of P.C. Newell.

Schmickl and Crailsheim (2007b) proposed a new method for communication and navigation within swarms of agents inspired by trophallactic behaviour exhibited by honeybees. In their method however, the receiver agent can query about the "nutritional" or fitness value of the local surrounding agents. From the nutritional value, the agent knows the gradient and can decide whether to go up-hill or down-hill.

Garnier *et al.* (2005) modelled a control algorithm of collective decision such can be seen performed by group of cockroaches. The control algorithm itself is based on small simple set of behavioural rules as follows:

- Random walk behaviour in the centre of arena, with constant rate of changing direction

and forward oriented distribution of turning angles.

- Wall following behaviour when reaching the periphery of the arena, with constant rate to leave the edge and come back towards the centre of arena.
- Stop at any moment under the shelter (dark place), stay motionless for some time and then move again.
- Stop for some time if perceive any other agent within perception range, stay motionless for some time and then move again.

They (Garnier *et al.* 2005) successfully implemented their control algorithm on the Alice micro-robots, with two shelters (dark place) placed in the arena. Based on their observations, the behaviours of the agents were similar to the behaviour exhibited by cockroaches, in which the cockroaches tend to aggregate in the dark area. Moreover, if there are more than one shelter available in the arena, cockroaches will usually collectively choose one of the shelters to be the aggregation location.

Kodati *et al.* (2007) designed and fabricated a micro underwater robot, named MARCO. MARCO gains inspiration from the boxfish (see Figure 2.10), this being is highly stable and fairly manoeuvrable. With multiple fins, the boxfish can manoeuvre in confined spaces with near zero turning radius. Furthermore, it has been found that the boxy shape is responsible for self correcting mechanism that makes its trajectories immune to water disturbances. It is believed that MARCO will be able to be used in many applications such as environmental monitoring, ship wreck exploration, inline pipe inspection, forming network sensor and so on.

Zhang *et al.* (2007) constructed another biologically inspired fish-like robot. They designed the robot to be able of propelling itself through oscillations of a flexible caudal fin, like a real underwater fish. The caudal fin is driven by a unique actuator called electrostatic film motor. At the current state, their robot achieves fish-like manoeuvring and approximate velocity of 0.018 m/s in dielectric liquid.

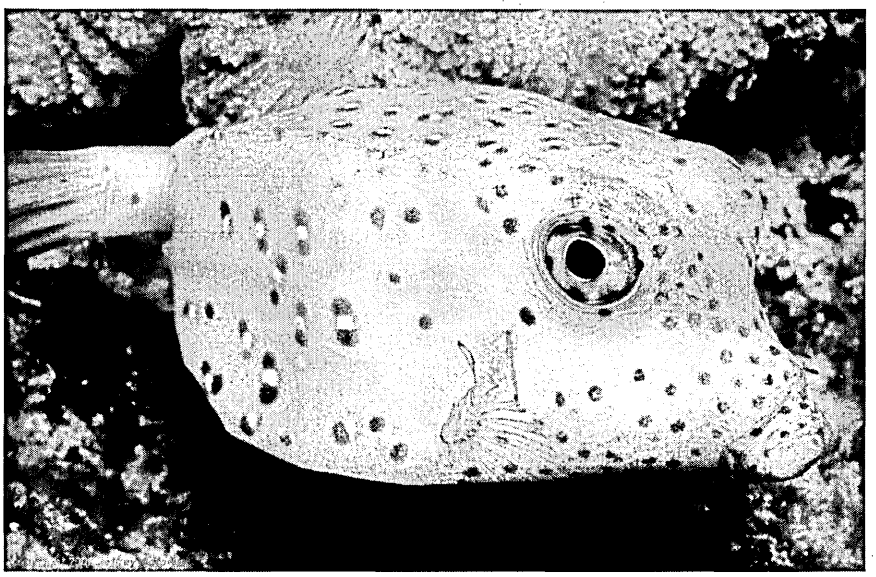


Figure 2.10: A boxfish. MARCO (Kodati et al. 2007) the under water robot gains inspiration from boxfish.(Image courtesy of divegallery.com)

Shao *et al.* (2006) introduced situation-based action selection mechanism for multiple fish-like agents to achieve cooperative transportation task. In this approach, each agent has an ID and the agent will do the task if and only if the conditions are met. The only problem with this approach is that, there is no overlap in the rules of the conditions. If one of the agents fails, the cooperative task will not be succeeded.

2.5.2 L-Systems

Lindenmayer Systems (L-Systems) is one of the many branches of Alife. Traditionally L-Systems have been widely used in the modelling of branching structures and the growth process of biological objects such as plants and micro-organisms. As technology advances, L-Systems have attracted more and more researchers from many diverse fields. Most researches on L-Systems concentrate on the modelling of plant growth or modelling the growth of multi-cellular organisms. However, in the following we review some works in other fields.

Hornby and Pollack (2001) used L-Systems and evolutionary algorithms to create a variety of virtual creatures. Their system made use of L-Systems to encode the creature and an evolutionary algorithm engine to evolve the creature. The creatures evolved by the system consisted of hundreds of parts. The end-product are "natural looking" creatures. The

components that make up the creatures include bar structure, every single type of joint, chain structures *etc.* Once a set of creatures are created, they will be evaluated by the evolutionary algorithm engine to find "fittest" creature. The fitness of the creature is evaluated based on the distance moved by the creature's centre gravity. Rolling and stepping are permitted but dragging imposes a penalty. Their results show that, over half of the simulation runs are able to generate interesting results and the most common creature movement involves rolling sideways. Another interesting movement is that of an undulating sea-serpent, like an inch-worm.

Zamir (2001) formulated parametric L-Systems to generate branching structures that can embody the physiological laws of arterial branching. He gives an example by showing that a complete cast of the arterial system of a rat can be modelled using parametric L-Systems as in Figure 2.11. From the results, it was suggested that the parametric L-Systems can be used to produce fractal like tree structures. However, the branching structures' parameters generated differ slightly with the variability in branching parameters observed in arterial trees. The parameters include the asymmetry ratio, the area ratio, branch diameters, and branching angles. The main issue in generating branching structures of arterial branching is that the source of variability in those parameters is not known, thus, it cannot be accurately reproduced in a model. Finally, he concludes that the L-Systems with a random choice of parameters can be made to mimic some degree of the observed variability, but the legitimacy of that choice is not clear.

Mariano *et al.* (1995) used L-Systems to generate large instances of the Euclidean Travelling Salesman Problem (ETSP). They gave 4 examples and successfully showed how L-Systems can generate patterns or paths for the ETSP. The patterns used in their work are MNPeano, MPeano, Koch and David's star. However, the method has a drawback where the distance between each of the two cities has to be the same.

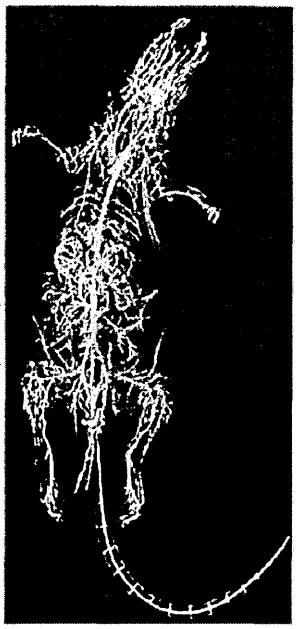


Figure 2.11: A complete cast of the arterial system of a rat. Modelled by parametric L-Systems (Zamir 2001).

Salvador *et al.* (2002) proposed the multi-fractal network traffic model based stochastic L-Systems. Their work consisted of an alphabet of arrival (packet) rates which is defined by:

$$\vec{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_L\}, \quad \lambda_i \in \mathbb{R}_0^+, \; i = 1, \ldots, L \tag{2.6}$$

and with production rules that randomly generate two arrival rates from a previous one. Without loss of generality, they made an assumption that $\lambda_1 < \lambda_2 < \ldots < \lambda_L$. From the real data observed, the L-Systems parameters are fit by the fitting procedure. It starts by fixing a sampling interval Δ and considers the time series representing the total number of packet arrivals in each sampling interval. The inference process in this model can be divided into three steps:

- determination of the L-System alphabet and axiom,
- identification of the time scale ranges, and
- inference of the L-System production rules.

From their numerical results, that include applying the fitting procedure to real observed data with multi-fractal scaling behaviour showed that, L-System based models achieved an excellent fitting performance.

Kókai *et al.* (1999) used parametric L-Systems in the GREDEA (Grammatical Retina Description with Evolutionary Algorithms) system. GREDEA is a system to develop patient specific monitoring programs for examining the blood circulation of the retina, which can be used on patients with diabetes who need to be monitored over long periods. At the beginning, the retina of a patient is scanned with laser ophthalmoscope (SLO). Then a parametric L-System is developed in which will create the pattern closest to the vascular tree of the patient's retina. The main reason for the L-Systems used here is because the L-Systems needs less storage than storing a picture.

Other examples of the Alife have been described in the Organised Formations section in this Chapter.

2.6 Swarm Modelling

In modelling swarms, many mathematical models were proposed by biologists to gain insights into the nature of swarming behaviour (Parrish *et al.* 2002). Most of the models proposed are focused on the spatial model, where space is directly or indirectly considered within the model (Gazi & Passino 2004). In the spatial model, Parrish *et al.* (2002) suggested that there are three distinctly different approaches that have been used to model the swarm dynamics; namely *Eularian*, *Lagrangian* and behaviour-based model (Reynolds 1987)(Grunbaum & Okubo 1999).

2.6.1 Eularian model

The first model is based on the statistical model and uses the *Eularian* framework to describe the mean-field density of swarm. In this approach, Edelstein-Keshet (2001) modelled the swarm as a density in spatial space by a partial differential equation that is based on a diffusion approximation of the random motion. Mogilner and Edelstein-Keshet (1999) extended the model by integrating non-local interactions, such as visual or auditory sensing. Although the model invites many analytical results that can be produced, the model is however limited to large and dense swarms with no big discontinuities (Parrish *et al.* 2002).

2.6.2 Lagrangian model

The second model is based on individual-based path generation, where *Lagrangian* equations are used to describe the motion of each individual member in the swarm (Gazi & Passino 2004). In this model, all attractions amongst individuals are modelled as attraction and repulsion forces. Furthermore, all attractions between individuals in the swarm can be modelled as potential functions, and the motion of each individual follows the negative gradient of the potential surface, which in turns serves as an attractive feature of this model. Moreover, by constructing a Lyapunov function which is associated with the potential force, the minimiser corresponding to the stable state of the swarm can easily been shown. Although the form of attraction / repulsion functions in this model are varied, it is understood that the aggregation is caused by the long-range attraction and the short-range repulsion (Couzin & Krause 2003)(Mogilner *et al.* 2003). For instance, Mogilner *et al.* (2003) proposed a model where attraction and repulsion terms were exponential with different magnitudes. With the model, they derived the individual distance of a large group, which in the end revealed a condition on the attraction and repulsion to avoid dispersion of swarm.

2.6.3 Behaviour-based model

The third spatial approach uses a behaviour-based model. In this approach, no explicit mathematical equations are proposed, and all interactions amongst individual agents are described by some behaviour rules. In the field of swarm engineering, Reynolds (1987) was one of the first to simulate behavioural control animation. He developed a system to model flocking behaviour and coordinated movements seen in birds and fish in which he named the creatures as ***boids***. The basic Reynolds' flocking model is based on three simple steering behaviours, namely *cohesion*, *separation* and *alignment*, which describes how an individual boid should change its heading or direction and velocity based on the positions and velocities of its nearby neighbours or flockmates. It is worth noting that in some literatures, the rules are also known as *flock centring*, *collision avoidance* and *velocity matching* which refer to *cohesion*, *separation* and *alignment* respectively.

Figure 2.12 shows three basic strategies of Reynolds' flocking rules. The circles indicate sensing range for the boids in the centre. This means that the boid in the centre of the circle can see or senses other boids within the circle. From the left is the cohesion, separation and alignment strategies respectively. Cohesion strategy as shown by the red boid on the left in the Figure 2.12, the boid feels the urge to steer towards the average position of its flockmates in its vicinity, resulting in the boids staying close to one another.

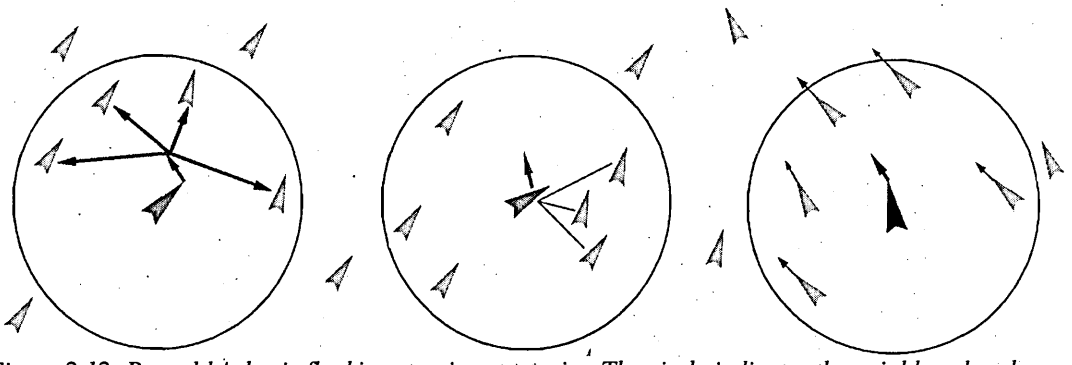


Figure 2.12: Reynolds's basic flocking steering strategies. The circle indicates the neighbourhood range of the agent's in the centre of the circle. The left shows cohesion, the centre shows separation, and the right shows alignment strategy respectively.

The green boid in the centre of Figure 2.12 exhibits the separation strategy; this strategy is to ensure that the boid is maintaining a safe distance from its flockmates and encourages the boid population to avoid crowding the neighbourhood. Finally, the blue boid on the right of Figure 2.12 demonstrates the alignment strategy which sometimes is referred to as the velocity matching strategy. This rule encourages the boid to move with a similar heading and velocity as its flockmates.

By using Reynolds's model of boids, Tanner *et al.* (2003a, 2003b) investigated the algebraic graph theoretical properties of underlying interconnection graph between agents. They also showed the relationship between the graph connectivity and stability of the flocking behaviour in fixed and dynamic topology.

There is one other similar work to Reynolds's flocking model, which was developed by Viscek *et al.* (1995). Viscek proposed a simple model in which each agent's heading is updated at

every time step as the average of headings of the agent itself and its nearest neighbour plus some additive noise. By comparing Viscek's model with Reynolds's model, it can be concluded that Viscek's model is a special case of Reynolds' model, where all agents move with same velocity, only following an alignment rule and only considering the nearest neighbour as a flockmate. From the results of their simulations, they (Viscek *et al.* 1995) showed that their agents move in a coherent manner, in which the headings of all agents converge towards a common value.

Folino and Spezzano (2002) adopted Reynolds's flocking rules and proposed a parallel spatial clustering algorithm for swarm agents called SPARROW (**SPA**tial Cluste**R**ing Algo**R**ithm Thr**O**ugh S**W**arm Intelligence). The algorithm combined a smart exploratory method based on a flock of birds with a density-based cluster algorithm to discover clusters of arbitrary shape and size in spatial data.

The motion of each agent follows the Reynolds's flocking model. Furthermore, SPARROW considers types of agents, grouped on the basis of the density of data in their neighbourhood. To differentiate the different types of agents, different colours are used as shows in Figure 2.13 below; red, showing a high density of pattern in the data, green a medium one, yellow a low one, and white indicates a total absence of patterns. The main idea of SPARROW is to take advantage of the coloured agents in order to explore more accurately in the tight cluster regions and avoid the ones without clusters. In simulation, the red and white agent will stop moving in order to signal out the type of regions, whilst the green and yellow agents will flock and move toward dense cluster. In this algorithm, the agents behave like hunters with a foraging behaviour that allow each agent to explore the spatial data while searching for cluster with different sizes, shapes in noise data (Folino & Spezzano 2002); cluster with different densities (Folino *et al.* 2003) in 2-D space. Moreover the algorithm also works in multidimensional space (Augimeri *et al.* 2006).

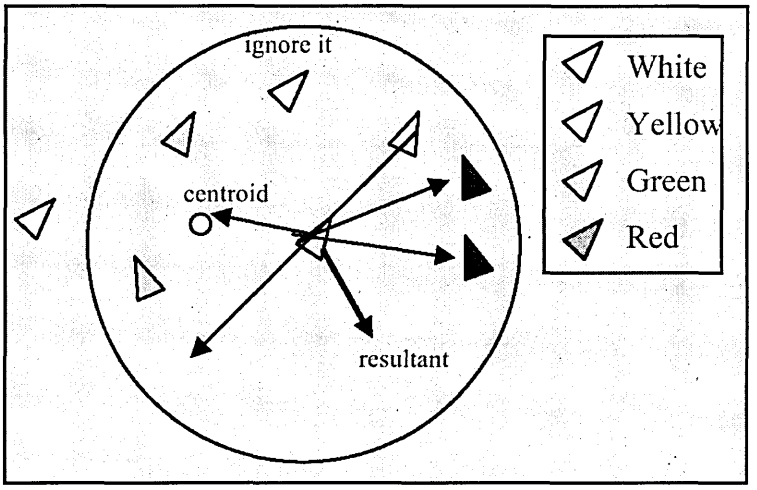


Figure 2.13: Cohesion strategy in SPARROW, taken from Folino and Spezzano (2002). In this strategy, the green agent in the centre feel the attraction towards red agents, and repulsion against white agent.

Olfati-Saber and Murray (2003) modelled a net flocking framework in the presence of multiple obstacles in the arena. Net-flocking is where agents in the flock will have the "bonds" between them when they come within close proximity of each other. Agents will keep this bond and stay close to each other. The bond will break if bonded agents move apart more than the allowed distance. The easy way to view net-flocking is to think the agents as the "atoms" and these atoms are connected to each other with these "bonds". They showed that the flocking behaviour is achieved by dissipation of energy according to a protocol that only requires the use of local information. The three basic flocking rules of Reynolds' are hidden inside this protocol. They defined three types of agents which are called alpha, beta and gamma. These agents are then used to create, what they call, net-flocking. They (Olfati-Saber & Murray 2003) also showed that by using their framework, the *split*, *rejoin* and *squeezing manoeuvres* flocking while avoiding obstacles can be done.

Some other works on swarm modelling that are worth mentioning were researched by Levine *et al.* (2000), Toner and Tu (1998) and Shimoyama *et al.* (1996). Levine *et al.* (2000) created rotating swarm agents known as circular *ant mills* using a self-propelled particles based model in which each agent can interact with all other agents in the arena. They modelled the flock in one- and in 2-D, and showed that the density of the flock drops to zero at the edge, or the density of the flock has a sharp edge confirming the work done by Mogilner and Edelstein-

Keshet (1999). Toner and Tu (1998) analysed Viscek's model and used a continuum mechanics approach to model flocking behaviour. Shimoyama *et al.* (1996) proposed a mathematical model of flocking and clustering motion such as collective rotation, chaos and wandering. They also categorised the behaviours and characterised the transitions of the models.

2.7 Simulation Tools

In developing and simulating multi-agent systems or swarm robotic systems, there are a number of purposely built computer programs available and ready for use. In this thesis three different software packages have been used to simulate our swarming algorithms; these are the Breve, Netlogo and MATLAB simulation tools.

2.7.1 Breve

The Breve toolkit (Klein 2002) was developed by Jon Klein during his year at Hampshire College, USA as a thesis project, and was developed further into a Master's thesis at Chalmers University of Technology, Sweden. Breve is also actively being developed as a platform for a project building large scale simulations of evolutionary dynamics, and many other applications.

Breve is a free simulation environment distributed as an open-source software with contributions from researchers from all over the world. It is designed for the simulation of multi-agent, 3D spatial and physical systems. It allows users to observe the interactions of predefined behaviours of autonomous agents in a continuous 3D world.

The world in Breve is represented as a 3D space and is able to facilitate 3D spatial simulations as shows in Figure 2.14 below. Agents in the simulation can occupy this 3D space to move around and about and interact in the 3D space. Breve allows the agents to be spatially aware and to comply with physical laws, therefore making the simulations closer to the real world.

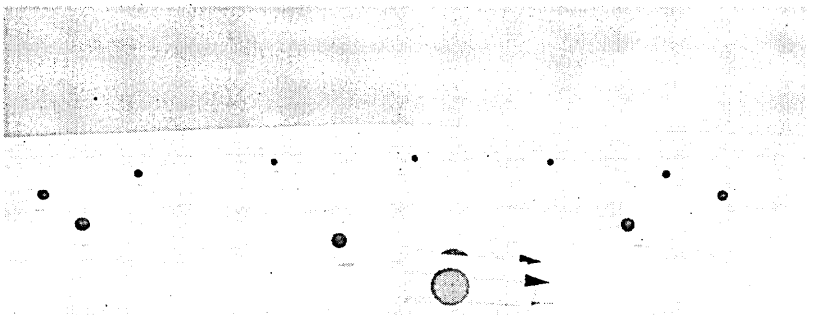


Figure 2.14: Example of Breve simulation world; showing the simulation of Braitenberg vehicle written by Klein (2002).

By enabling Breve's main feature of physical laws, one can simulate breve agents to behave just as real objects do, according to the laws of physics. For example, if an agent is defined as a ball and placed in the air above the floor, the physical simulation engine will make the ball realistically fall towards the floor and bounce back, subject to gravitational forces and Newton's Third Law. Amongst other things, the physical simulation engine in Breve can be used for realistic simulation of robots, vehicles and animals as well.

Breve simulations are usually written in an object-oriented and easy to use language called *STEVE.* The language borrows many features such as in C, Perl and Java. Breve also features extensible plugin architecture which allows programmers to write plugins and interact with pre-existing code and projects. In the simulations, all aspects including object and memory management, communication between agents, and integration are automatically handled by the Breve engine (Spector *et al.* 2005b).

Another main feature of Breve is the fact that Breve supports the *Push* programming language. Push was developed specifically for genetic programming and other evolutionary computational applications. Push is designed to avoid most of the complications that can arise when writing evolutionary codes. The two main characters of Push programming are that it has very unusual simple syntax and the ability to work flexibly with multiple datatypes (Spector *et al.* 2005a).

Spector and Klein have used Breve in many of their works, including their most notable work where they demonstrate the evolution of a form of multicellular organisation, and altruistic food sharing for flying agents (Spector *et al.* 2005b).

2.7.2 NetLogo

NetLogo was created in the spirit of the Logo programming language which is easy to learn, to use and to read, but also powerful enough to deal with complex concurrent problems. Logo was developed by a mathematician Seymour Papert in mid 1960s. At that time Seymour was working with the team from BBN (formerly known as Bolt, Baranek and Newman), led by Wallace Feurzeig. The first implementation of Logo was written in LISP (List Processing language) and released in 1967.

Logo was originally designed to introduce children to programming concepts and thus develop better thinking skills that could be transferred into other contexts. It was aimed to be enable easy entry by novices and yet meet the needs of high power users.

The most well-known Logo environments have involved the *turtle*. The turtle is originally a virtual creature that sits on the floor and could be directed to move around by receiving commands from a user or programmer. The turtle is used to draw shapes, designs and pictures.

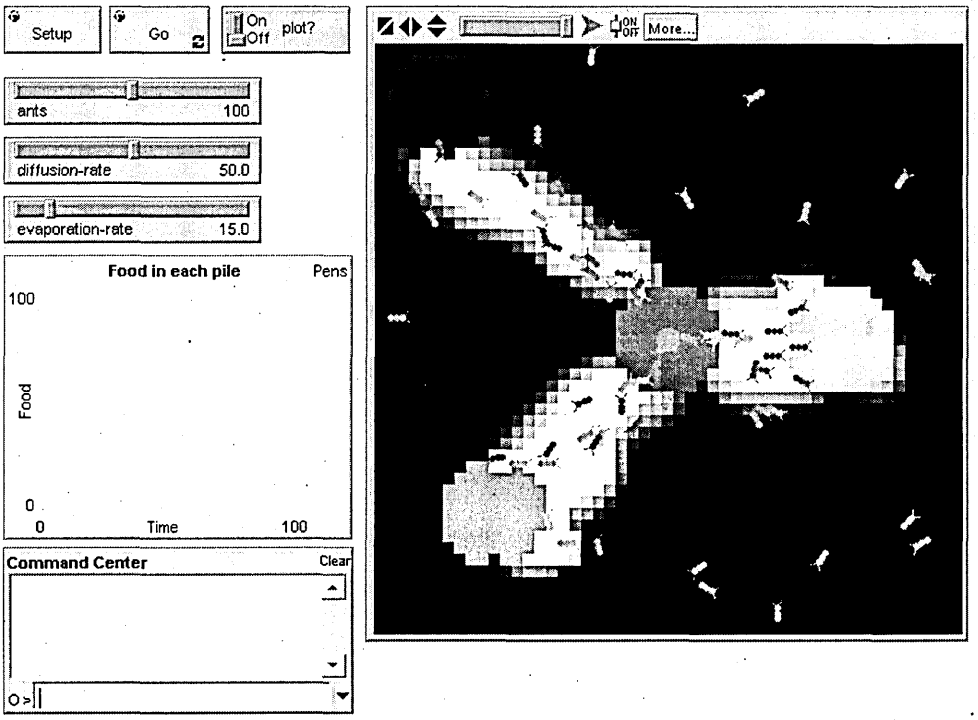


Figure 2.15: Example of NetLogo simulation world, showing the simulation of the ant foraging model written by Wilensky (1999).

NetLogo (Figure 2.15) was written and released by Wilensky in 1999. It was originally developed at the Centre for Connected Learning and Computer-Based Modelling at Northwestern University and is in continuous development at the Center for Connected Learning and Computer-Based Modelling at the same university.

NetLogo is well suited for modelling time-dependent complex systems and literally allows users to give instructions to hundreds or thousands of independent agents operating concurrently. This makes it possible to explore the connection between micro-level individual behaviour and macro-level patterns that emerge from the interactions of individuals.

NetLogo is specifically designed for deployment of models over the Internet and is written in Java so the model can be run on all major operating systems (NetLogo 2008). After five years of development, NetLogo is a mature simulation tool which is stable and fast (Tisue & Wilensky 2004).

Extensive documentation, tutorials and demonstrations are available on the package's website. NetLogo comes with a Models Library which contains a large collection of more than 140 pre-written simulations code that can be used and modified. These pre-written code in models library includes wide range of disciplines and education levels; from natural to social sciences, mathematics and computer sciences.

Even though NetLogo is distributed as a freeware, the functionality in NetLogo can easily be extended through an Application Programming Interfaces (APIs).

There has been a considerable amount of work completed on multi-agent systems modelling using NetLogo. One of them is the work done by Momen *et al.* (2007). In this work, they have modelled two species of "birds" and studied the effect of multi-species flocking. These two species of birds will attract to each other depending on the *heterospecific-attraction* parameter. From the results, they showed that as the heterospecific-attraction increases, the flocking efficiency also increases. Another model using NetLogo is the work by Veeraswamy *et al.* (2006), where they promoted the use of path planning with the ant foraging technique. Results of their simulations showed that the performance of the ant foraging problem can be improved dramatically by combining the regular ant foraging algorithm with the A* path planning algorithm.

2.7.3 Other Simulation Tools

A number of other simulation tools exist that are used by the swarming community.

MASON, the Multi-Agent Simulator of Neighbourhood was developed by a joint effort of Evolutionary Computation Laboratory (ECLab) and Center for Social Complexity of George Mason University, USA (MASON 2008). MASON is written in JAVA to take advantage of its portability, operating system independence, object serialisation and strict math and type definitions (Luke *et al.* 2005). It is designed to be used for a wide range of simple simulation with emphasis on swarm multi-agent simulations.

MASON is released as an open-source in which users are free to use and modify the source code. At present, there is little documentation and it has a relatively small user group. However, some of the documentation detailing of how-to use and some of the publications detailing the implementation or application of MASON are available for a prospective user to evaluate further (MASON 2008).

Webots™ is a proprietary software and developed by Cyberbotics Ltd.. Cyberbotics Ltd. is a limited company derived from Swiss Federal Institute of Technology in Lausanne (EPFL) and was founded in 1998.

Webots™ provides a rapid prototyping environment for modelling, programming and simulating mobile robots. The robot libraries enable users to transfer control programs to many commercially available real mobile robots such as Khepera™ , Aibo™ and the LEGO™ Mindstorms™ robots. Webots™ offers numerous features to make the simulation tool easy to use and able to do complex computations (Michel 2004). The features include:

- allowing the user to model and simulate any mobile robot, including legged, wheeled and flying robots
- allowing the user to program the robots in C or C++ or JAVA, or from third party software through TCP/IP
- using the Open Dynamic Engine (ODE) library for more accurate physics simulation
- many examples with controller source code and models of commercially available robots
- creating AVI or MPEG simulation video file for online or public presentations.

Many other multi-agent simulators exist, such as Gazebo, Player / Stage, Repast and so on. A more comprehensive review on multi-agent simulators can be found in Castle & Andrew (2006) and Railsback *et al.* (2006) and the references therein.

2.8 Summary

The pretext of this Chapter is aimed at trying to understand the conceptual and natural roots surrounding pattern formation in swarms. Various state of the arts methods related to the subject have been reviewed. This lays out the ground work for the original contributions that will be presented in the following Chapters.

A number of popular robotics control architectures were reviewed and described in detail. These include the well known Brooks's subsumption architecture (1986) and Reynolds' flocking algorithms (1987). Research work in autonomous mobile agents or robotics field has since the early days suffered from difficulties associated with centralised planning.

A review of control algorithms for distributed pattern formation for robot swarms have shown that the agents are more complex. In Particular, agents need communication modules (such in Avrutin *et al.* 2007, Payton *et al.* 2004, Nouyan *et al.* 2006, Freeman *et al.* 2006, Desai 2002, Fierro & Das 2002, Kaminka & Glick 2006, Pavone & Frazzoli 2007, *etc.*), or the ability to perform complex calculations (such in Yang *et al.* 2007, Desai 2002, Takahashi 2004, Mastellone *et al.* 2007, *etc.*), or the requirement for vision based sensors (such in Das *et al.* 2002) in order to carry out pattern formation tasks. With these levels of complexity, the (hardware) cost of building swarm agents increases significantly. In addition complex agents have a higher probability of failure to due to the integration of multiple crucial components. In keeping complexity down, agents will have minimal sensors and onboard processing power. The problem now becomes determining how relatively simple agents can be controlled. Thus it is a challenge to design pattern formation control algorithms on such a simple swarm agent.

In the review on the swarming behaviour, most of the work in the literature reported involving repelling and attracting factors or "repulsion and attraction forces" (such in Tanner 2003, Hanada 2007, Olfati-Saber & Murray 2003, Esposito & Dunbar 2006, Chen *et al.* 2007, Desai 2002, Mastelone *et al.* 2007, Yang *et al.* 2007, *etc.*). These repelling factors are used in agent-to-object and/or agent-to-agent interactions so that agents will not collide with each other and/or with another object (obstacle avoidance strategy). On the other hand however, the

attracting factor only accounts for agent-to-agent attractions which is usually used in a strategy such that agents will remain close to each other, for example the cohesion strategy as in (Reynolds 1987). The problem of the attracting factor for agent-to-object interaction still needs to be examined. It will be interesting to investigate how agent-to-object attraction forces will affect the behaviours of swarming agents.

With regard to the bridge formation connecting two objects or two locations (such as in Avrutin *et al.* 2007, Nouyan *et al.* 2006) an important issue is that to be able to connect the two locations, the proposed algorithms require a large number of agents. A particular problem arises when only the minimal number of agents that are required to make a bridge formation are present in the arena. For example, if there are twenty agents in the arena and a minimum of twenty agents are needed to form a certain bridging formation in the arena, the proposed algorithms might take a long time to find a solution without additional agents. Worst still it might not be able to find the solution at all within the permissible time frame. It is then useful to devise an alternative algorithm even if it is not fully automated or self organising for the reason that self-organisation is not always the best solution for every problem.

The next three Chapters describe the overall methodology by which the research was carried out including developing areas identified in this Chapter.

Chapter 3 State Based Models

The aim of swarm engineering is to design multi-robot platforms that are able to mimic biological robot swarms in performing tasks where a group of robots, each of which has limited capability, can perform better than just the one. In simulating large scale swarms, computational cost plays an important role, thus limiting and stagnating development. Many studies have been undertaken using a practical approach to swarm construction. Amongst these are studies investigating navigation and exploration tasks, task allocation, elementary construction, and communication.

In this Chapter, two control algorithms using the Finite State Machine (FSM) approach are developed to support simple swarm robots in swarm robotics pattern formation, where complex behaviours can emerge from interactions between agents and each agent with the environment. This Chapter proposes that by alternatively switching on and off a combination of transmitters and sensors of agents, different variety of agent behaviours can be achieved. The work here is loosely motivated by ants which have limited memory and limited ability and yet they are able to form a line to and fro from their nest to a food source. In the first algorithm the agents are tested with forming chains or lines, and in the second they are tested with forming a cluster. In both cases agents have very little memory, limited sensing capabilities and processing power, there is no explicit communication between agents and the formations are formed based solely on environment cues.

3.1 Introduction

The ability to self-organise in a predicted way is important for the accomplishment of a wide range of tasks in the swarm robotics domain, thus many different approaches have been proposed (such as in Avrutin *et al.* 2007, Payton *et al.* 2004, Nouyan *et al.* 2006, Freeman *et al.* 2006, Desai 2002, Fierro & Das 2002, Kaminka & Glick 2006, Pavone & Frazzoli 2007, *etc.*). Each agent in the swarm system is capable of performing very simple tasks, but when these agents aggregate, they are able to act as a larger entity which is able to perform more complex tasks. To achieve this, initial studies have to be undertaken in order to understand the act of swarming by biological systems and inferring rules that govern the swarm movement.

The research in this Chapter was conceived during the initial development phase of the EU FP6 I-SWARM project. Several initiatives have been undertaken to create, understand and simulate swarming behaviours. Due to the limited capability of the I-SWARM agents, which directly affects the number of sensors that can be mounted on them, and their limited processing capabilities, novel methods for encoding and processing information have to be developed. Moreover, due to the size of the agent and to reduce the manufacturing costs, communication modules are not present on the agents, thus excluding the ability to communicate explicitly like agents in multi-robot systems. Agents have very little memory and limited sensing capabilities which are used to detect obstacles and other agents.

In this piece of research, the main interest is in studying and implementing rules that lead to basic swarming behaviour on very simple agents. In the field of swarm robotics, which emphasises the cooperation and collectivity of groups of agents, individual agents are usually controlled by simple strategies. Complex behaviours are often achieved at the group or colony level by exploiting local interactions amongst agents, and its environment. In designing control algorithms for swarm agents, often complex strategies are avoided. Instead, simpler principles such as homogeneity of agents and distributiveness of control algorithms are preferred.

There are many related works in swarming that are already mentioned in Chapter 2 (Section 2.6, page 54). However, the control algorithms proposed in these works require high agent

complexity in term of hardware (such in Das *et al.* 2002, Yang *et al.* 2007, Desai 2002, Takahashi 2004, Mastellone *et al.* 2007, Avrutin *et al.* 2007, Payton *et al.* 2004, Nouyan *et al.* 2006, Freeman *et al.* 2006, Desai 2002, Fierro & Das 2002, Kaminka & Glick 2006, Pavone & Frazzoli 2007, *etc.*), which is the luxury that the agents within this research do not have. For examples, agents in (Das *et al.* 2002) need vision-based sensor; in (Yang *et al.* 2007, Desai 2002, Takahashi 2004, Mastellone *et al.* 2007, *etc.*) need to perform complex calculations hence requiring large amounts of memory and processing power; and in (Avrutin *et al.* 2007, Payton *et al.* 2004, Nouyan *et al.* 2006, Freeman *et al.* 2006, Desai 2002, Fierro & Das 2002, Kaminka & Glick 2006, Pavone & Frazzoli 2007, *etc.*) the agents require a fairly complex communications module installed, thus defeating the aim of swarm intelligence.

With the aforementioned constraints, two particular pattern formation tasks that have been addressed in this research are how swarm agents can be coaxed into forming a line or chain and how agents can be programmed to cluster in a bounded arena under the constraint that agents have a limited memory, sensing ability and processing power. This work is loosely inspired by the observation of ant colonies, but unlike ants these agents do not release pheromones in order to attract other ants. Rather agents use infra-red transmitters to attract other agents to themselves.

3.2 Tasks and Approaches

With the limitation of agents as previously mentioned, the FSM approach has been chosen in this part of the research. The tasks that have been chosen for the collection of relatively simple agents to perform are line formation and cluster formation. The agents do not have any prior knowledge about the dimensions of the working arena or how many other agents are present in the arena. These other agents can also be viewed as dynamic obstacles in the environment. The sensing range is small relative to the working arena. In this research, a reactive decentralised control algorithm has been investigated and used to perform the abovementioned tasks. The mission is to complete the specific task while avoiding collisions with other agents and the wall.

In the following section, the simulated world is first described, followed by the agent design and its dynamics. Next the control mechanisms will be described at both low and high levels. The results of the simulations are then presented and discussed.

3.3 Simulation Environment

3.3.1 Simulator and agent designs

The experiments presented in this Chapter have been conducted in a physical simulation engine called Breve (Klein 2002), first introduced in Chapter 2. Breve is specifically designed for the simulation of multi-agent, 3-D spatial and physical systems. Hence agents will be subjected to normal Newtonic laws. As shown in Figure 3.1, the working arena consists of a *floor* which is defined as a cube of 70 by 70 patches with thickness of 5 units in the Breve world. Along the perimeter of the *floor*, the *wall* has a thickness and height of 4 units. The wall is included to prevent agents from falling off the end of *floor*.

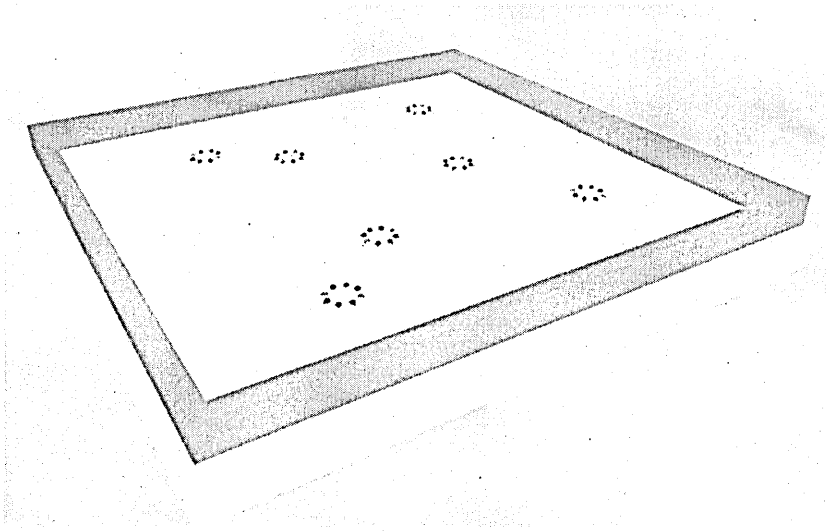


Figure 3.1: Simulation world; showing floor, wall with seven agents in the arena.

Swarm systems can either be homogeneous or heterogeneous systems. In a homogeneous swarm system, agents usually consist of physically identical agents with exactly the same hardware and software capabilities. Whilst on the other hand, in a heterogeneous swarm system, the agents may be different, such as at the hardware or software levels. Even if the

agents have exactly the same hardware configurations and control software, if each agent has a unique identifier, the swarm system is considered as a heterogeneous swarm system. Hence in a homogeneous swarm system, each agent is identical in all respects.

Within this study, several homogeneous physical embodied agents with homogeneous and heterogeneous control have been designed to examine a number of swarming algorithms. As shown in Figure 3.2, each agent is installed with a ring of 8 equally spaced infrared transmitter-receiver pairs around its turret, that enables the agent to attract and detect other agents from its local environment. In addition to transmitter-receiver pairs, the agent has two driving wheels and two omnidirectional wheels, which allow the agent to move in any arbitrary direction and step once it is commanded to do so.

The agent has been defined as a mobile *multibody* object. The body of the agent, which has been labelled as *RoboBody* has been defined as a root body or root link. The *RoboBody* then has been connected to other parts of the agent, such as wheels and sensors. The *RoboBody* has been constructed using a *PolygonDisk* object and has been defined with a radius of 2 units with thickness of 0.75 units and with sides of 40 units. The wheels have been created using the *PolygonDisk* object as well with a radius of 0.60 units, thickness of 0.21 units and sides of 40 units. The wheels are referred to as *leftWheel* and *rightWheel* to differentiate the left and right wheel of the agent.

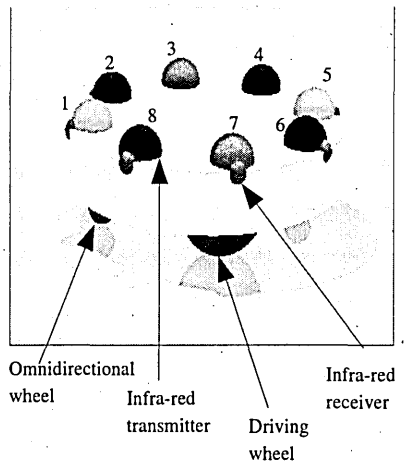


Figure 3.2: Simulated agent

The *leftWheel* and *rightWheel* are then assembled to the left and right side of the undercarriage of *RoboBody*. To be exact, the *leftWheel* is connected to *RoboBody* at the point of (0, -0.30, -1.20), and the *rightWheel* is at (0, -0.3, 1.20) as shown in Table 3.1.

The joints between the wheels and *RoboBody* has been defined as a revolute joint so that the wheels can be controlled to roll backward and forward just like a physical wheel. The omnidirectional wheel supports have been constructed using spheres with radii of 0.20 units. The supports have been placed at the front and the back of *RoboBody*. The front support, *frontSupport* is connected to *RoboBody* at the point of (1.5, -0.68, 0), and the back, *backSupport* support is at (-1.5, -0.68, 0) as shown in Table 3.1. The joints between omnidirectional wheel and *RoboBody* has been defined as a ball joint so that the wheel can rotate freely in 360 degrees.

Eight pairs of infra-red transmitters and receivers have been created. The transmitters and receivers have been defined as spheres. The pairs are then assembled at the top and near the end side of the RoboBody spaced at 45 degrees apart as shown in Figure 3.2. The transmitters and receivers faces outward and are perpendicular to the side of the *RoboBody*.

Table 3.1: Agent's parts

Agent's parts	Shape	Link	Joint type	Connected to RoboBody at
RoboBody	PolygonDisk with: sides = 40, thickness = 0.75, radius = 2	Primary	None	Not available
leftWheel	PolygonDisk with: sides = 40, thickness = 0.21, radius = 0.60	Secondary	Revolute joint	(0, -0.3, -1.2)
rightWheel	PolygonDisk with: sides = 40, thickness = 0.21, radius = 0.60	Secondary	Revolute joint	(0, -0.3, 1.2)
frontSupport	Sphere with: radius = 0.2	Secondary	Ball joint	(1.5, 0.6, 0)
backSupport	Sphere with: radius = 0.2	Secondary	Ball joint	(-1.5, 0.6, 0)
transmitters	Sphere with: radius = 0.3	Secondary	Fixed joint	various
receivers	Sphere with: radius = 0.15	Secondary	Fixed joint	various

These infra-red transmitter and receiver pairs are labelled 1 through 8 in the clockwise manner from the front position of the agent to differentiate which one which. These pairs are used for attracting and locating other agents in the environment. The transmitters can be switched on and off individually.

The lime, blue and red colours of the transmitters as in Figure 3.2, are used to indicate that those transmitters are switched on. When the transmitter is switched off, white is used. The chosen colours are also used to assist the observer to recognise in which direction the agent is facing. As indicated in Figure 3.2, red is on the left and right side and lime is at the front and back side of the agent. For each pair, the infra-red receivers are placed in front of the transmitters, this is to avoid the receiver from receiving the signal transmitted by its own pair.

The transmission and detection range of transmitters and receivers has been set to twice the radius of *RoboBody, i.e.* 4.0 units. The transmission angle for transmitter is set to 25 degrees, whilst the detection angle is 45 degrees.

The collision sensor is located at the front of the *RoboBody* just below the infra-red receiver number 1. The sensor range is set to 4.0 units and the detection angle is set to 60 degrees.

3.3.2 Agents dynamics

The locomotion of the agents are non-holonomic, where there is a restriction on the maximum possible turning angle of the agents. Turning is achieved through two driving wheels located just below transmitters 3 and 7. The movement of wheels are governed by the input receivers.

At the beginning of the simulations, the natural velocity of wheels have been set to 2.50 rad/s. The natural velocity is the speed at which the wheels turn in the absence of sensors or receivers input. Each wheel can be controlled individually by changing the value of left and / or right wheel velocity, namely *leftSpeed* and *rightSpeed* respectively.

In this study, several flags for the agent's movement have been defined. These include *speedUp*, *slowDown*, *speedNorm*, *speedStop*, *turnRight* and *turnLeft* as shown in pseudo codes in Table 3.2. At any one time, only one or none of the flags will be set to true while the others are set to false.

From Table 3.2, when *speedUp* is switched to logic true, both *leftSpeed* and *rightSpeed* will increase their value based on their current value by 1.3 times. The *speedUp* method will be activated if the distance between the agents is greater than 3 unit. The increment of *leftSpeed* and *rightSpeed* values will stop when either the current value of *leftSpeed* or *rightSpeed* is reached or is greater than 3.0, or the *speedUp* flag is set to false. The *speedUp* method is useful when an agent is far from the other agents and is trying to keep close to the others within the allowed distance.

The *slowDown* method is the opposite of *speedUp*. The *slowDown* method will be activated if the distance between agents is less than 2 unit and greater than 1 unit. Whenever the *slowDown* flag is set to true, the *leftSpeed* and *rightSpeed* parameters will be reduced by half until it reaches the value of zero or the flag is set to false, whichever comes first. The *slowDown* method has been used in the simulations whenever the agent comes to close contact to other agent(s) or obstacle[1] before comes into a halt.

Whenever an agent needs to turn to the right or left, the flag *turnRight* or *turnLeft* will be set to true accordingly. As shown in the Table 3.2, during the *turnRight* method, the *rightWheel* will turn to the opposite direction of *leftWheel*, resulting in a negative value of the *rightSpeed.* The *turnLeft* method on the other hand, will set the *leftWheel* turning in the opposite direction.

The movement of an agent can be stopped by setting the *speedStop* flag to true. This will set the *leftSpeed* and *rightSpeed* to zero, resulting in the agent coming to a halt. The *speedStop* method will be activated if the distance between agents is less than 1 unit. The last elementary flag as shown in Table 3.2 is called *speedNorm*, this method will be activated if the distance between

1 In this Chapter, even though there no obstacle has been defined, agents treat the perimeter wall as "obstacle".

agents within 2 and 3 unit. In this method, both wheels will be set to *naturalVelocity*, resulting the agent will move on to straights line forward.

Table 3.2: Pseudo code for agents dynamics

```
while speedUp do
        if leftSpeed < 3.0 and rightSpeed < 3.0 then
                leftSpeed = leftSpeed * 1.3
                rightSpeed = rightSpeed * 1.3
        end if
end while

while slowDown do
        if leftSpeed > 0 and rightSpeed > 0 then
                leftSpeed = leftSpeed * 0.3
                rightSpeed = rightSpeed * 0.3
        end if
end while

if turnRight then
        leftSpeed = leftSpeed * 1.0
        rightSpeed = rightSpeed * -1.0
end if

if turnLeft then
        leftSpeed = leftSpeed * -1.0
        rightSpeed = rightSpeed * 1.0
end if

if speedStop then
        leftSpeed = 0
        rightSpeed = 0
end if

if speedNorm then
        leftSpeed = naturalVelocity
        rightSpeed = naturalVelocity
end if
```

3.4 Encoding of Rules

Within this research, since agents in the final I-SWARM agents have a limited amount of memory, the rules governing the motion of each agent (in this Chapter) are based on a behaviour based architecture, consisting of three states represented by finite state automatons as shown in the state diagram of Figure 3.3. Each state corresponds to a different behaviour. At each *simulation time step* only one behaviour is active. The transitions between agents are dependent on the sensory inputs which is represented by s in the Figure 3.3; where $s = 0$ means there is no sensory input whilst $s > 0$ means there is at least one of the agent's sensor gives a positive reading.

To test the hypotheses of pattern formations of robotic swarm using state models, two control algorithms have been proposed, namely *line formation* and *clustering*. In line formation, all agents have been controlled homogeneously, whilst in clustering, agents have been controlled heterogeneously. For clustering, two types of agents are defined, that is *attractor* and *searcher* agents which use different control sets.

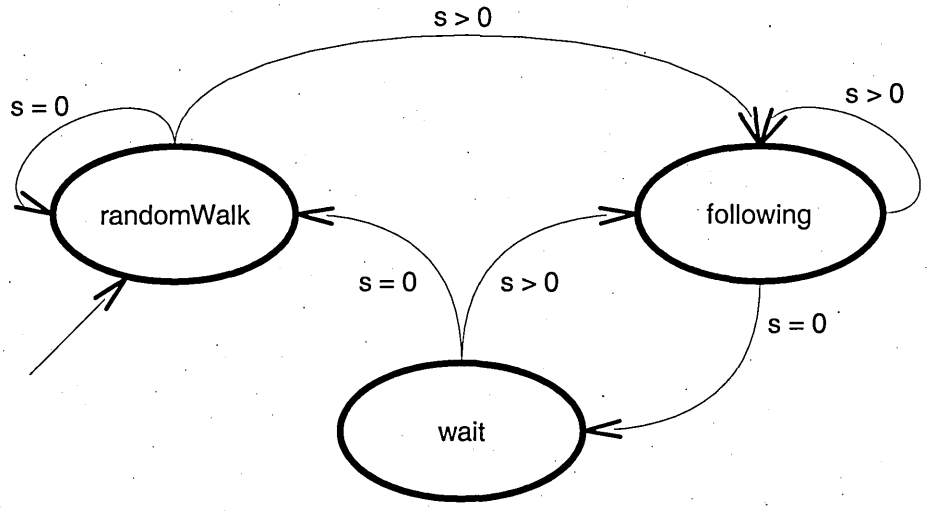


Figure 3.3: State diagram for line and cluster formation. States are shown as labelled circles while transitions are depicted as arrows. Each transition is labelled as event *which triggers the transitions. Letter s represents* sensory input *to the agent; where* s = 0 *depicts no sensory input, while* s > 0 *means there at least one of the sensor gives a reading.*

The approaches for *line formation* and *clustering* are thus described. First the behaviours are described, followed by the conditions that trigger the transition between behaviours for all agents, *i.e.* in *line formation* and *searcher* agents in *clustering*, and finally a high-level descriptions for *line formation* and *clustering* control algorithms is provided.

The three behaviours designed into the agent are:

- randomWalk: agent performs a random walk in the arena looking for others within its vicinity. All infra-red transmitters are switched on for *line formation* and all are off for *clustering*.
- following: move towards the agent in front. Back three infra-red transmitters (numbered 3, 4 and 5) are switched on, others are off for *line formation*; front three (numbered 1, 2 and 8) are off while others are on for *clustering*.
- wait: moves with the current wheel speed. On-off arrangement for infra-red transmitters are same as in the *following* behaviour.

The behavioural transitions are:

- randomWalk → following: if agent perceived another agent. Note that an agent only can be perceived by other agent if and only if its infra-red signal is detected by the other agent.
- following → wait: if an agent lost the infra-red signal that has been detected before.
- wait → randomWalk: if an agent perceived the agent that was lost previously.
- wait → randomWalk: if an agent could not detect the lost infra-red signal within permitted time frame.

3.4.1 Line formation

At the beginning of the simulation, agents are placed at the predefined location as shown in Figure 3.4. Typically, an agent will not detect any other agents in its vicinity and is in the random walk state. The movement of the agent while in *randomWalk* state is that at every simulation time step the agent will have a small probability of 0.005, or one in 200 chances to

turn its direction of heading. This is to avoid agents moving in a Brownian motion; moreover it will give agents the ability to scan a wider area in a shorter period of time. When the agent needs to change its direction, it will choose randomly either to turn left or right by either 45 or 90 degrees.

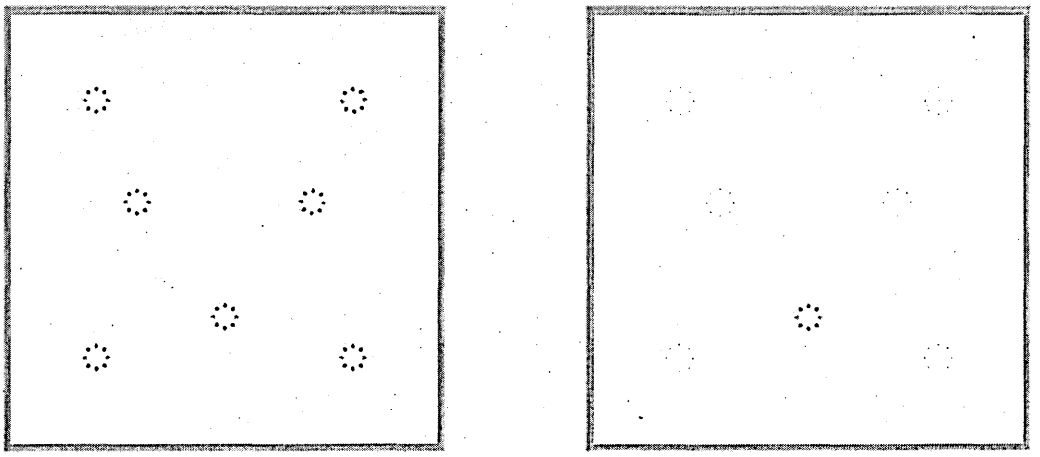


Figure 3.4: Agents start position in the arena for; (a) line formation, (b) clustering.

While in the *randomWalk* state, the agent will switch on all its transmitters as shown by the agent on the right in Figure 3.5 to attract others to its positions. At the same time, the agent will look around and tries to find for external infra-red signals from another agent through its receivers and within its sensitivity range.

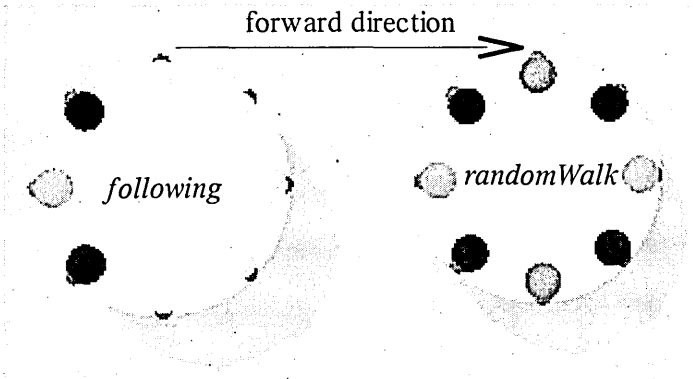


Figure 3.5: Transmitters' on-off arrangement for line formation. Agent on the left is in the following state where three transmitters at the back is switched on. On the right is in the randomWalk state where all transmitters are on.

As shown in Table 3.3 algorithm 1, during the *randomWalk* state, the agent will read each of the receiver values until it gives the logic true, *i.e.* the agent perceives another agent. Reading the values of the receivers is done sequentially, from the front to the rear and from right to left. In other words, the value from the receivers will be read by the following order of receiver's number: 1, 2, 8, 3, 7, 4, 6 and finally 5. The sequential reading of sensors is in line with the limited processing capability of the on-board microcontroller of many swarm robots.

Table 3.3: Pseudo codes for three states in line formation algorithm.

Algorithm 1. *randomWalk()*	Algorithm 2. *following()*	Algorithm 3. *wait()*
switch-on all transmitters **for** (each receiver) **do** **if** (any receiver) *following()* **end if** **end for**	switch-on transmitters (4,5,6) switch-off transmitters (1,2,3,7,8) **for** (each receiver) **do** **if** (any receiver) *following()* **end if** **if** (all receivers == false) reset counter *wait()* **end if** **end for**	**while** (counter < 50) **do** **for** (each receiver) **do** **if** (any receiver) *following()* **else** counter ++ **end if** **end for** **end while** **if** (counter >= 50) *randomWalk()* **end if**

As soon as one of the receivers detects the existence of another agent within its neighbourhood, it will fall into the *following* state and make the necessary turn towards the agent. The agent will also ignore the receivers that are reading while turning or changing its heading. For example, if the receiver numbered 7 detects the signal, the agent will turn 90 degrees to the left and ignore the reading from receivers until the turning task is done. Likewise for receiver number 3, it will turn to the right by 90 degrees and ignore the readings from receivers until

completed the turning task.

In the *following* state, the agent switches off all its transmitters except the three to the rear, numbered 4, 5 and 6, as the agent on the left shown in Figure 3.5. The reason behind this is to attract another agent to its rear in order to form a line.

There is no explicit communication taking place between the agents. Agents rely solely on their receivers to control the movement. This kind of behaviour is known as cue based behaviour where agents react to stimuli in its environment. As shown in Table 3.3 algorithm 2, during the *following* state, as in the *randomWalk* state the agent will read each of its receivers values sequentially from the front to the rear and from right to left until one of the receivers detects the infra red signal from the agent in front. If the signal is detected, the agent will remain in the *following* state.

Due to the fact that infra-red transmitters and receivers have a limited effective transmission and detection range and angles, a *wait* state has been introduced. This is to prevent the agent from moving to the *randomWalk* state from the *following* state directly it loses the signal by a few degrees. This can happen when the agent in front is turning or changing its heading by a few degrees. During this *wait* state, the agent will keep moving forward without turning left or right with its current wheel speed. As in the *following* state the agent will only switch on the rear three transmitters, numbered 4, 5 and 6

As can be seen in the algorithm 2 of Table 3.3, the counter is reset to zero every time the agent exits the *following* state and enters the *wait* state. In the *wait* state, the agent will read each of its receivers values sequentially as in *following* and *randomWalk* state.

While in the *wait* state (algorithm 3 of Table 3.3), if the agent detects any infra-red signal from any other agent, it will change its state to the *following* state, and make a necessary turn towards the front agent. If the agent does not detect any signal, it will increase the counter by one and continue to read the receivers values.

This continues until the agent detects the wanted signal or the counter reaches (or exceeds) the value of 50, whichever comes first. If the agent detects the signal, it will change its state to the *following* state, otherwise it will return to the *randomWalk* state.

3.4.2 Cluster formation

For *cluster formation* amongst agents, two types of agents namely *attractors* and *searchers* have been predefined. The *attractor* agent will act as a leader and will try to attract *searcher* agents to its position. The pseudo codes and state diagram that govern the *searcher* agents movement are shown in Table 3.4 and Figure 3.3 (in page 77) respectively.

During the simulation, the *attractor* agent switches on all its transmitters permanently in order to attract *searchers* to gather around its position. It also moves randomly in the arena until it bumps into other agents several times, and it will stop at that position.

Table 3.4: Pseudo codes for three states in cluster formation algorithm.

Algorithm 1. *randomWalk()*	Algorithm 2. *following()*	Algorithm 3. *wait()*
switch-on all transmitters **for** (each receiver) **do** **if** (any receiver) *following()* **end if** **end for**	switch-on transmitters (3,4,5,6,7) switch-off transmitters (1,2,8) **for** (each receiver) **do** **if** (any receiver) *following()* **end if** **if** (all receivers == false) reset counter *wait()* **end if** **end for**	**while** (counter < 50) **do** **for** (each receiver) **do** **if** (any receiver) *following()* **else** counter ++ **end if** **end for** **end while** **if** (counter >= 50) *randomWalk()* **end if**

As shown in the state diagram in Figure 3.3 (in page 77), the searcher will begin the simulation in the *randomWalk* state. While in *randomWalk* state, the *searcher* agent will switch off all its transmitters and it will move randomly in the arena while looking for others.

At every simulation time step, *searchers* in the *randomWalk* state and the *attractor* will have a small chance with a probability of 0.005 to turn its direction of heading. When turning, the agent will choose randomly either to turn 45 or 90 degrees to the left or right.

As shown in Table 3.4 algorithm 1, during the *randomWalk* state, the *searcher* will read each of its receivers until any infra-red signal from any other agent is detected. Reading is done sequentially from front to rear and from right to the left as in the line formations method discussed earlier.

Once the *searcher* detects any infra-red signal, the *searcher* will move into the *following* state and turns towards the detected signal. While in the *following* state, the *searcher* will switch on all its transmitters except the three at the front as shown by the agent on the left in the Figure 3.6. By doing so, it will attract other searchers to its back or side in which will form a cluster of agents in the end.

During the *following* state (Table 3.4, algorithm 2), as in the *randomWalk* state the agent will read each of its receivers values sequentially from front to the rear and from the right to the left until one of the receivers detects the infra red signal from the agent in front. If the signal is detected, the agent will remain in the *following* state. Otherwise the agent will move to the *wait* state and reset the *wait* counter to zero.

In the *wait* state (Table 3.4 algorithm 3), the *searcher* agent will keep on moving forward with its current wheel speed and on-off arrangement of its transmitters. The *searcher* agent will also read its receivers value as before. If any of the receivers detects any infra-red signal, it will move back to the *following* state and make the necessary turn as in the line formation.

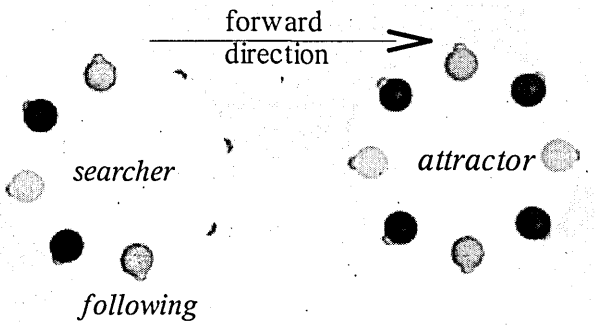


Figure 3.6: On-off arrangement of transmitters for cluster formation. On the left is the searcher *agent in the following state switching on all the transmitters except the front three. Agent on the right is the* attractor *switching on all its transmitters.*

If the *searcher* agent is not able to detect the wanted signal, it will increase the counter by one and will search again for the signal. The process will loop until the searcher detected the signal or the counter reaches the value of 50 or greater, whichever comes first. If the signal has been found, the searcher will move back to the *following* state, otherwise it will move to the *randomWalk* state.

3.5 Experiments

3.5.1 Simulations setup

As previously mentioned, the research in this Chapter has been undertaken in conjunction with the I-SWARM project; where agents have little memory, limited sensing capabilities and no communications module installed on them. The goal of the simulations was to evaluate the controllers under the most basic conditions. In particular, we placed no obstacle in the working environment and agents are placed at the same position and orientation at the start of each simulations as shown in Figure 3.4 (page 79). In these simulations, seven agents have been used. We employ a bounded arena of size 70x70 units in the Breve world, as mentioned previously, for all the simulations. Fifty one runs are made for each control algorithms. The performance was evaluated at the end of the simulations and all runs for *line formation* and *clustering* were executed for 300 and 200 simulation seconds respectively to provide enough time for all agents to complete the task. The simulations were recorded into a movie format, and the data for analysis were recorded at every 20 simulation seconds.

3.5.2 Evaluating line formation

In evaluating each control algorithm, first the number of agents that are in the *randomWalk* state are counted. As the number of agents in the simulations was fixed at seven, the number of agents in the *randomWalk* state towards the end of the simulation is ideally one. When this happens, the agent which is in the *randomWalk* state will act as a leader for other agents. In other words, agents in the working arena will follow the agent ahead of itself, in the end will result a moving queue.

The reason for counting the number of agents in the *randomWalk* state is that if we suppose that each of the agents in *randomWalk* state acts as a leader for line formation, then the number of leaders in the environment will represents the number of lines or chains that formed in the arena. In this study this number should be minimal *i.e.* one; this will show that in the arena there is only one leader and one line or chain amongst the agents have been established.

Figure 3.7 shows the plot of the mean number of agents that are in the *randomWalk* state against time over 51 simulations run. As can be observed from the plot, the number of agents in the *randomWalk* state decreases over the first 100 seconds, and then stabilises afterwards. This shows that agents are able to form a line or two within the first 100 seconds. The low variation of the standard deviations demonstrates the consistency of the algorithms.

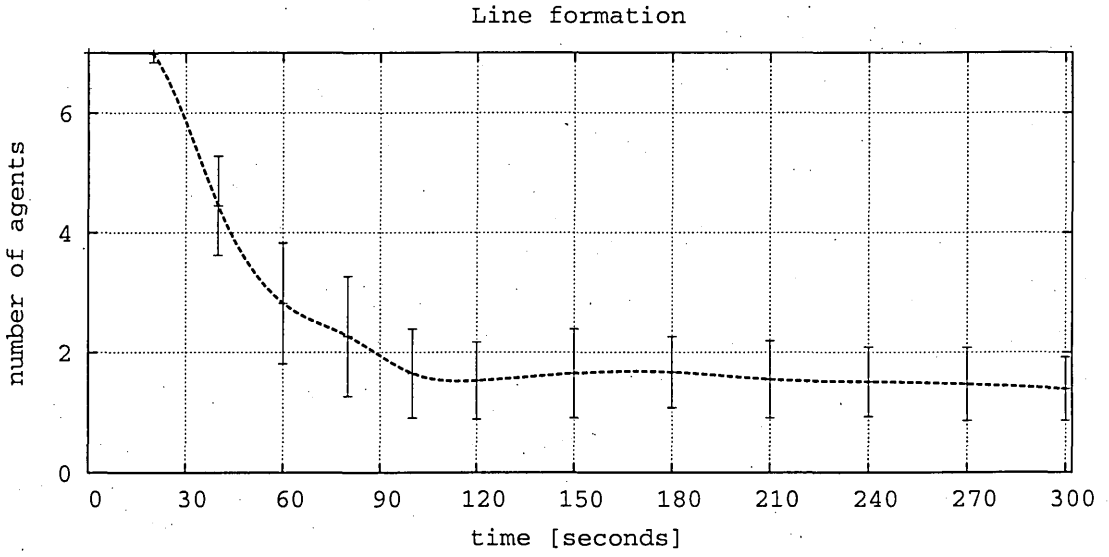


Figure 3.7: Number of agents in the randomWalk state in line formation against time.

In order to further understand how the agents converge, Table 3.5 shows the percentage of the number of agents in the *randomWalk* state over 51 simulations run. As the simulations began, all the agents fall into the *randomWalk* state which amounts to 100% if there are seven agents. At t = 20 seconds, only 1.96% or in other words, only once in the entire simulations run that one of the agent was not in the *randomWalk* state. Towards the end of the simulations, specifically from 150 to 300 seconds as we can see from the plot and the data provided, around 90% of the simulation runs have managed to form two lines or less, in which more than half of the time only one agent is in the *randomWalk* state and acts as a leader for the entire agents. From the observations during the simulations, there are a number of times where agents were successfully formed a single line and then split into two lines due to the limited angle of transmissions and receivers. We believe that this can be avoided by increasing the counter for the *wait* state.

Table 3.5: Percentage of number of agents in the randomWalk *state over 51 simulations run for line formation.*

Time / Agents	0	20	40	60	80	100	120	150	180	210	240	270	300
1	0	0	0	7.84	21.57	49.02	54.9	49.02	39.22	52.94	52.94	58.82	62.75
2	0	0	0	31.37	45.1	39.22	37.25	39.22	54.9	39.22	43.14	35.29	35.29
3	0	0	9.8	37.25	19.61	9.8	7.84	9.8	5.88	7.84	3.92	5.88	1.96
4	0	0	47.06	17.65	11.76	1.96	0	1.96	0	0	0	0	0
5	0	0	31.37	5.88	1.96	0	0	0	0	0	0	0	0
6	0	1.96	11.76	0	0	0	0	0	0	0	0	0	0
7	100	98.04	0	0	0	0	0	0	0	0	0	0	0

Consider the snapshots taken during one of the simulation runs for line formation in Figure 3.8 at 40, 100, 190 and 240 seconds respectively. In Figure 3.8(a) and (b), three of the agents have detected other agents hence have moved into the *following* state. Other agents which are in the *randomWalk* state will remain in the state until it detects another agent which they can follow. Figure 3.8(c) shows two of the agents are in the the *randomWalk* state and as can be seen that the agents have formed two lines; one with two agents and the other with five agents. Figure 3.8(d) shows that towards the end stage of the simulation which shows the agents have successfully formed a line.

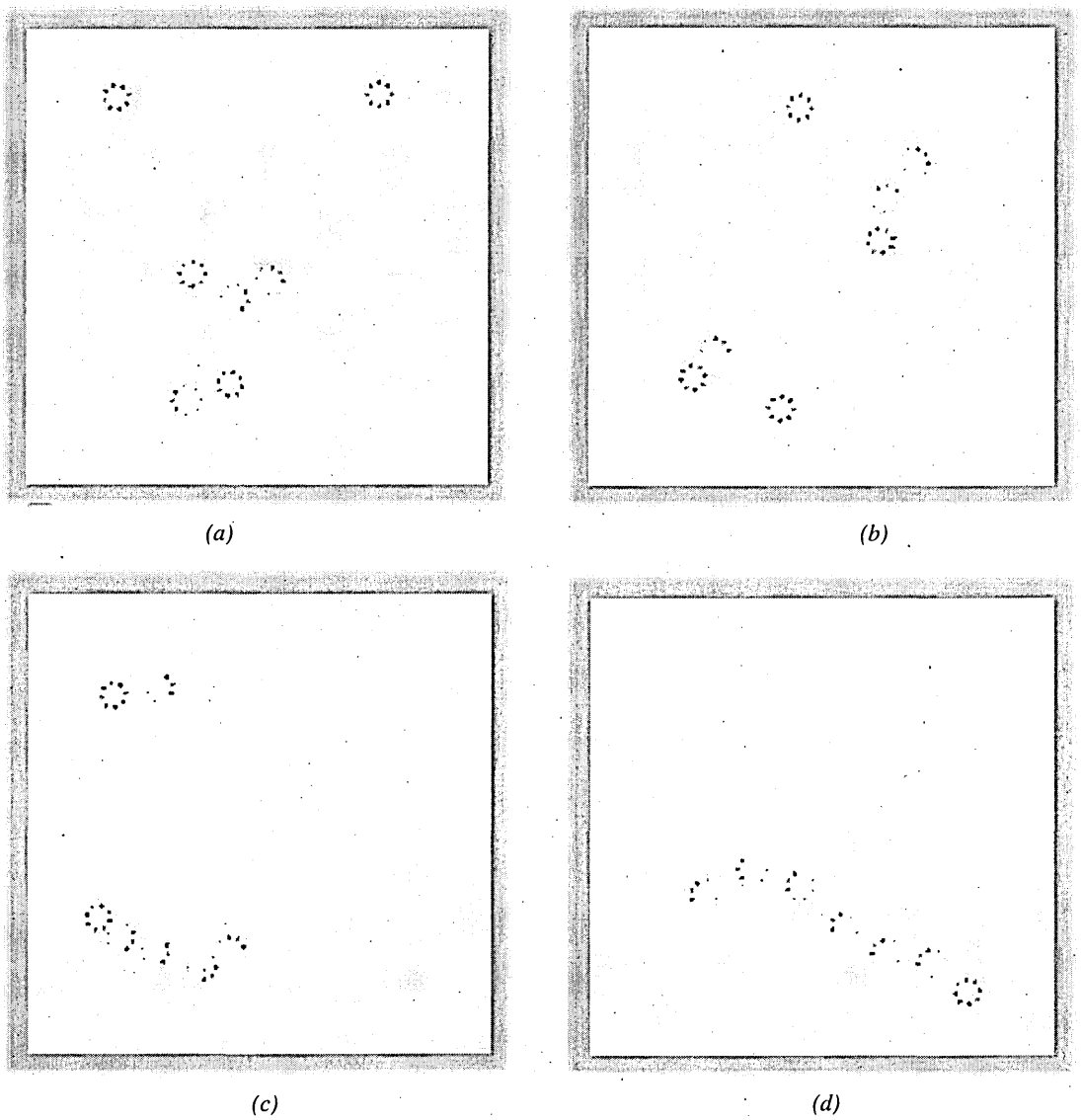


Figure 3.8: Stages in line formations: (a) at t = 40[s]; (b) at t = 100[s]; (c) at t = 190[s], and (d) at t = 240[s].

3.5.3 Evaluating cluster formation

In cluster formation, as mentioned previously two types of agents namely *searcher* and *attractor* agents have been defined. The evaluation process is similar to those in the line formation, but this time we take *searcher* agents into account and counted the number of *searchers* that fall into the *following* or *wait* state. In the simulation, we have a total of seven agents, six of them are *searchers* and one is an *attractor.* As cluster formation implies, the task for the *searcher* agents is to roam in the working arena and look for and gather around the *attractor.* As the number of *searchers* is fixed to six, ideally towards the end of the simulation all *searchers* found the *attractor* and the number of *searcher* agents in the *following* state is six.

Figure 3.9 is the plot of mean number of *searcher* agents that are in the *following* or *wait* state against time over 51 simulations run. As can be seen from the plot, from the start of simulations up to around 70 seconds, the number of agents changes rapidly. The rate of convergence seems to slow down after approximately 70 seconds and stabilises after 140 seconds.

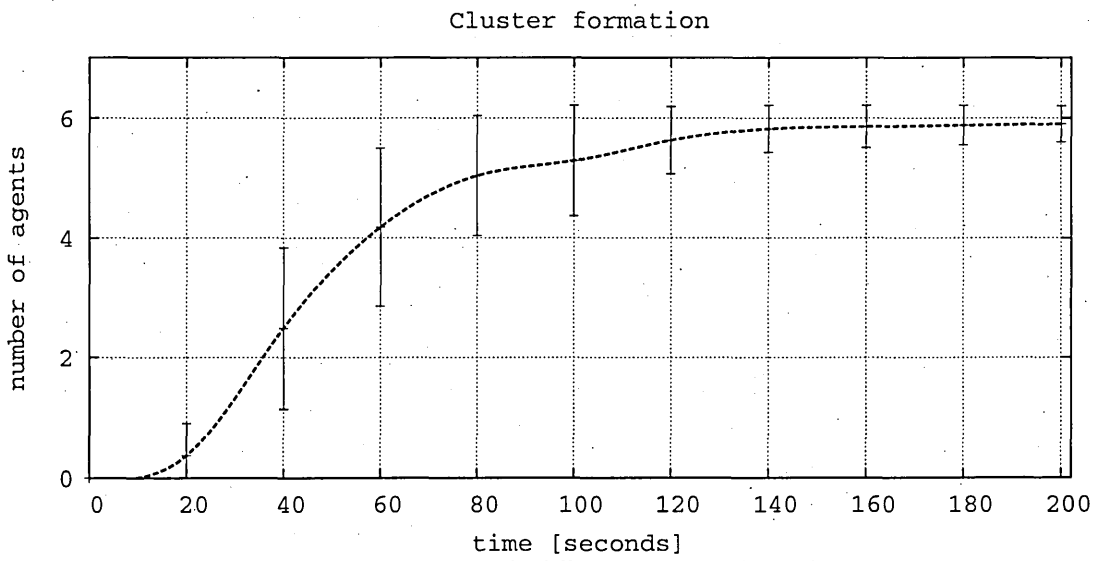


Figure 3.9: Number of searcher agents in the following state in cluster formation against time.

Table 3.6 gives an overview of the percentage of the number of *searchers* in the *following* or *wait* state over 51 simulations run. At the beginning of the simulations, none of the *searchers* are in the *following* or *wait* state, giving 100% to number of *searchers* 0. At 40 seconds, the number of *searchers* increases, but for most of the simulation runs only three or less of the *searchers* were in the *following* or *wait* state. Also at 40 seconds, only once from the entire simulation runs that all the *searchers* are already in the *following* or *wait* state, or have already completed the task of clustering. From 140 seconds, at least five of the *searchers* are in the *following* or *wait* state and more than 80% of the simulations run the task has been completed.

Table 3.6: Percentage of number of agents in the randomWalk *state over 51 simulations run for cluster formation.*

Agents \ Time	0	20	40	60	80	100	120	140	160	180	200
0	100	64.71	5.88	0	0	0	0	0	0	0	0
1	0	33.33	19.61	3.92	0	0	0	0	0	0	0
2	0	1.96	23.53	7.84	3.92	1.96	0	0	0	0	0
3	0	0	29.41	13.73	1.96	1.96	0	0	0	0	0
4	0	0	15.69	33.33	17.65	13.73	3.92	0	0	0	0
5	0	0	3.92	23.53	39.22	29.41	29.41	17.65	13.73	11.76	9.8
6	0	0	1.96	17.65	37.25	52.94	66.67	82.35	86.27	88.24	90.2

Figure 3.10 shows the snapshots taken during one of the simulations runs for cluster formation at different time stages. Figure 3.10(a) shows that at the beginning of the simulation, all agents were in the *randomWalk* state. The *attractor* permanently switches on all the transmitters, whilst the *searchers* switch off all the transmitters during the *randomWalk* state. Figure 3.10(b), shows that two of the *searches* have already encountered the *attractor* resulting in the *searchers* switching on the transmitters to its side and back. Figure 3.10(c) and (d) show the simulation runtime at 100 and 160 seconds respectively. At these times most of the *searchers* have perceived the *attractor* or other agent with the transmitters turned on. Finally Figure 3.10(d) shows towards the end of the simulation which shows all the agents successfully forming a cluster.

The strategy of the agent in the cluster formation simulation is the same as in the line formation algorithm, where the agent does not use any kind of explicit communication and relies only on its sensors or receivers to control its motion. Moreover, it is an auto-catalytic process, the more there are agents in the cluster, the larger the cluster becomes and the more likely other agents are to discover the cluster, thus reinforcing the growth of the cluster.

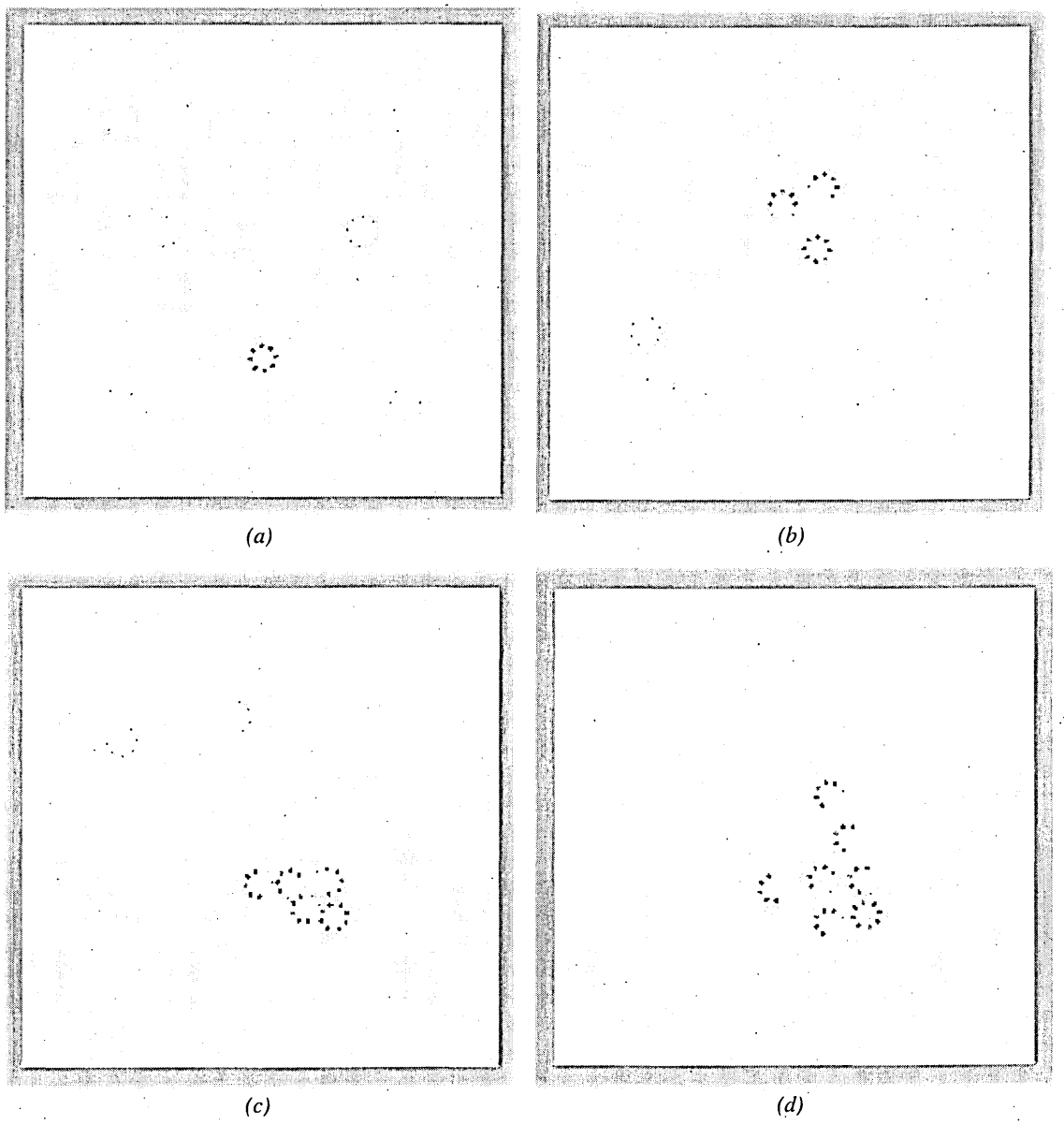


Figure 3.10: Stages in cluster formations in one of the simulation runs: (a) at t = 0[s]; (b) at t = 40[s]; (c) at t = 100[s], and (d) at t = 160[s].

3.6 Discussions

An experimental study of two simple control algorithms for pattern formation in robot swarms, using state-based and rule-based systems, have been presented. The agents are designed to be homogeneous in hardware which have constraints in processing power, little memory and limited ability, and it has been shown how the simple agents can be controlled in a homogeneous and heterogeneous way such that basic organisation can be achieved.

From the results, it is shown that by automatically switching on and off a combination of transmitters and sensors, a variety of agent behaviours can be achieved. It is also shown that simple pattern formation of mobile robot swarms can be obtained by using only simple rule sets without the need for any direct communication between agents.

The overall aim of this research was to investigate two different, but crucial problems in robot swarm. Firstly, the problem of self-organising in a robot swarm into an interesting pattern is a challenging task that has been studied by several research groups. Secondly, the potential for a swarm of robots to generate solutions that can meet real world constraint still remains to be achieved.

During this study, the major constraint that was identifies was the processing power and the on-board sensing capability of the robots. With limited capabilities, it might look that nothing substantial could be achieved by each individual agent. Hence, a way need to be found to overcome this issue.

The work discussed in this Chapter is related to achieving interesting and coherent behaviour from a number of simple agents. These simple agents only have little memory, limited sensing capabilities and processing power.

It is a much simpler task to design a controller for a robot that maximises its own sensing abilities, but the result is likely to be a very deterministic behaviour. By using the local interactions between robots, other information can be harnessed within the environment that is not necessarily directly available to all robots. This requires the agents to have the ability to perform localised signalling to their nearest neighbours. In this scenario, aggregation patterns are important for the flow of information within the robot swarm.

By having certain patterns encoded in the robot swarm, complex tasks can be more easily performed. Such patterns for example; line formation or a moving queue, can be useful in cleaning-type tasks, search and rescue tasks, optimal path finding between two points *etc.*

Another example is clustering, where the pattern will be useful for information sharing within the working arena, or to do some complex processing which could not be achieved by single agents due to limited local sensing and computational capabilities.

Another important area of investigation is the composition of several behaviours to produce more solutions in more complex scenarios. This way, a robot must use the limited sensing capabilities with some degree of context in order to "understand" its situation. This will give it the information it needs to make a decision on switching between rule sets. For example, using only its IR sensors, a robot can differentiate between an obstacle and another robot. If a robot picks up an IR signal, it can determine if it is reflected by an object by switching off its transmitters. Using this simple method, a robot can be part of a moving queue formation until discovering an object before taking action based on this information.

By combining localised signalling and context within a scenario, this work provides a step towards robot swarms being able to emulate complex dynamical pattern formations such as those present in nature, in social insects for example.

In this Chapter it has been shown how simple agents can be given simple rule sets to produce interesting behaviours. As each state within the state diagram is governed by rules to perform that automaton, so the resulting aggregated behaviour can be built upon to produce even higher levels of coordination. Ultimately what looks like a massive cooperation emerges from what are essentially local interactions.

From a practical approach, the work developed in this chapter has been used by Fernandez *et al.* (2005) to construct SHUBOTS at Sheffield Hallam University. The SHUBOT agents are shown in Figure 3.11. Each SHUBOT robot has low complexity and is low-cost, and so works ideally as swarm-capable agents to complement the work carried out in this Chapter. The SHUBOT consists of three modular platforms, namely: the microcontroller module, the sensor module, and the locomotion and powering module. The modular design approach was taken to allow for future possibilities of either expanding the platform or changing the sensor

configurations.

Fernandez *et al.* (2005) designed and studied various sensor combinations and presented three of the behaviours. The first behaviour is achieved when all transmitters / receivers combinations are switched on. In this behaviour, agents transmit its location and detects other agents at the same time. The behaviour is not deterministic due to:

- indefinite obstacle avoidance lock, due to the fact that since all transmitters are switched on, and when the agent encounters an obstacle and avoids it, it may again perceive the empty space as an obstacle, thus turning into the obstacle again.
- the breaking of robot chains, due to the fact that when an agent is following another agent in the vicinity the robot will rotate through by 180° when it gets too close.

The second behaviour is the leader-follower (line formation) behaviour as shown in Figure 3.11. In this behaviour an agent acts as the leader and other as followers, allowing for long chains to be formed. The leader switches off the back three receivers, so that it does not detect any follower. The leader roams and avoids obstacles. The followers on the other hand have all it's receivers and transmitters switched on, which allows it to detect a leader, or another follower agent that it can follow.

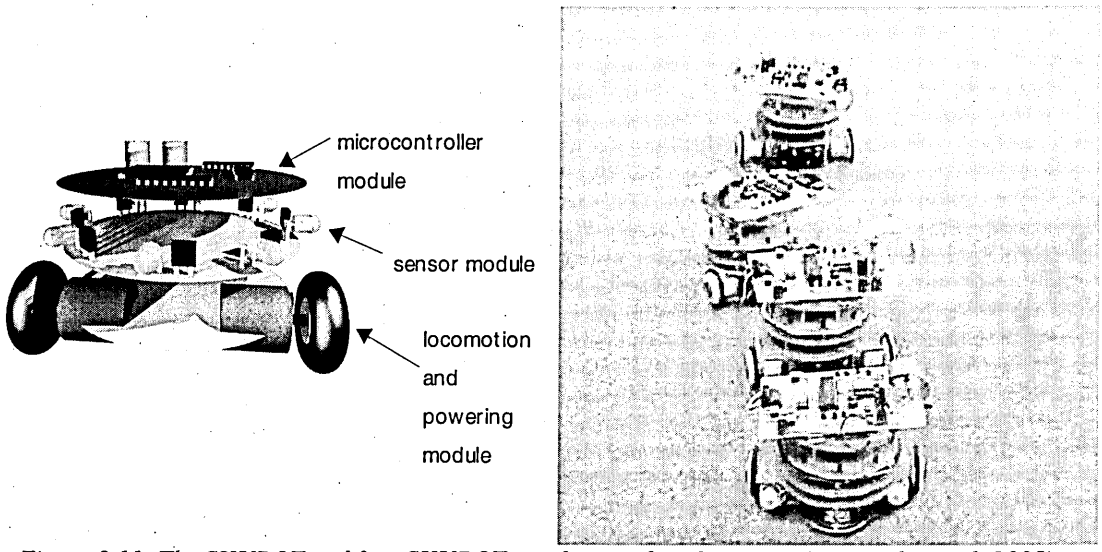


Figure 3.11: The SHUBOT and four SHUBOTs performing line formation (Fernandez et al. 2005)

The third behaviour is a clustering behaviour. In this behaviour, agents switch on the three frontal receivers, so other agent can be detected towards the front. When an agent is detected it follows it. Due to the fact that all transmitters are switched on, lateral following also occurs which results in a clustering behaviour.

Furthermore, their results (Fernandez *et al.* 2005) showed that the agents (SHUBOT) were able to distinguish between obstacles and other partner agents in the working environment. The method for doing so was to use a triple check approach as shown in Figure 3.12. It was also found that the agents encountered some difficulty due to multiple reflection from boundaries and a variety of infra-red sources, which has been ignored during the simulations. Nevertheless it was found that the simulations did provide a useful study in developing the physical agents.

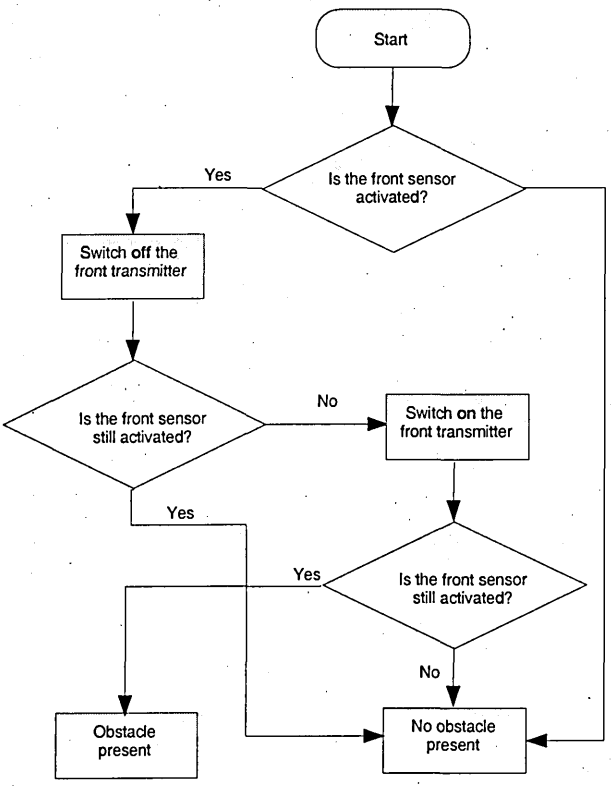


Figure 3.12: Obstacle avoidance on SHUBOT. The triple check is due to the fact that the robot's own infra-red transmitter may affect sensing.

To sum up, in this Chapter, two control algorithms using the FSM approach for pattern formation have been devised to support relatively simple swarm agents that have very little memory, limited sensing capabilities and processing power. It has been shown that even with relatively simple swarm agents, simple pattern formation of mobile swarm agents can be obtained by using only simple rule sets without the need of any direct communication between agents. In this work, different variety of agents behaviours are achieved by switching on and off a combinations of transmitters and sensors.

Chapter 4 Modelling of Collective Movement

Self-organising systems usually comprise a large number of autonomous and reactive agents where aggregations or collective movements are determined mainly by their neighbourhood influences. Generally these systems have been used to simulate and study natural and biological phenomena. With recent technological advances, the realisation of deploying hundreds (if not thousands) of swarm agents is becoming more viable. This Chapter examines how an artificial potential field affects the collective movement of swarm robots. In the next two sections the history and background of the flocking algorithms and collective movement in robotics are provided. Thereafter the simulation methodology and its implementation will be described. The results are evaluated and conclusions are drawn.

4.1 Introduction

In nature, there are countless examples where animals or insects gather in a large groups, displaying collective movement and self organise in a coherent fashion. These patterns are evident in numerous other examples of animal or insect migration behaviours such as the great herds of antelopes and wildebeest thundering across the Savannah in Africa, and Monarch butterflies migrating south from North America into remote mountain tops in central Mexico towards the end of summer days. The way that these appear coordinated and synchronised according to local rules is fascinating to discover.

It is hard to believe that for such a large group there does not exist a single entity or a leader to control the group's behaviour. For example, in the case of the birds flocking or fish schooling, the bird or fish at the front of the flock seems to lead, and the others to follow. On the contrary,

most bird flocks and fish schools are leaderless. In fact the movements of the flocks and schools are determined by instantaneous decisions of individual bird or fish.

Orderly flock patterns arise when each agent in the flock follows simple rules in response to dynamic interactions within neighbourhood. Such movements are a prime example of self organisation in swarms. Camazine *et al.* (2001) state that the main feature of self organisation is that a system's organisation or movement does not explicitly depend on external control factors. In other words, the organisation emerges solely due to the local interactions between individuals and their environment. The organisation also can evolve in either space or time and can maintain some kind of stable form or can show in transient phenomena. An example of such a system is that of a colony of ants sorting eggs without knowing any particular sorting algorithm (Bonabeau *et al.* 1999).

An example of self organisation in a swarm is the flocking of birds. As previously mentioned in Chapter 2 (page 57), Reynolds (1987) was one of the first to simulate flocking behaviours of birds. The basic Reynolds' flocking algorithm is based on steering behaviours of which he labelled as Separation, Alignment and Cohesion. The result of the simulations was a movement model that mimics various swarms in nature, a school of fish for instance. Since the flocking work of Reynolds (1987), there are many works which are related to and extended from the flocking or swarming algorithms. Wilensky (1999) for example, further developed the simulation inspired by the boids algorithm. The algorithm presented by Wilensky (1999) (described in the next section) is very similar to the original boids algorithm but not entirely the same.

Other works which were inspired by the Reynolds' include the work of Tanner (2003), Hanada (2007), Olfati-Saber & Murray (2003), Desai (2002), Mastelone *et al.* (2007) *etc.* Most of the works reported in articles involve "repel" and "attract" factor (such in Tanner 2003, Hanada 2007, Olfati-Saber & Murray 2003, Esposito & Dunbar 2006, Chen *et al.* 2007, Desai 2002, Mastelone *et al.* 2007, Yang *et al.* 2007, *etc.*). These repel factors are used in the agent-agent and/or agent-object obstacle avoidance strategy, meanwhile the attract factor is only used in the

agent-agent cohesion-like strategy. The work in this Chapter investigates and examines the problem of agent-object attraction factors by extending the flocking algorithm of Wilensky (1999). This will lay groundwork of how agent-object attraction factors affect swarms behaviour in performing an aggregation task.

4.2 Collective Movement in Robotics

Movements in mobile agents can be classified into two categories, holonomic and non-holonomic motion. Holonomicity in mobile agent refers to the relationship between controllable movement DOF (degree of freedom) and the total DOF of a given agent. If the controllable movement DOF is larger than total DOF, the agent is considered to be a holonomic agent.

For example, let us consider a mobile agent with two wheels, one on each side of the agent's body. Each wheel has two DOF which can be controlled to turn either clock- or anti-clock-wise, independently, and thus the agent has 4 controllable DOF. By having different directions (clock- or anti-clock-wise) and/or speed of the wheels, the agent can freely move on a planar surface with 3 physical DOF; hence the agent is a holonomic agent.

In multi-agent systems, each agent has to control its motion in order to form some degree of cohesive motion with other agents within the group. Methods for achieving collective and coordinated motion are dependent on the sensing and processing capabilities of the agent. Generally, the movement of agents are mainly reactive which is completely determined by reflexive movement dynamics. Interactions between agents and its dynamic environment will result in "complex" macroscopic behaviour and promote self organisation in the end.

In swarm robotics, collective movement is a very important aspect of many tasks. Often, agents have a limited sensing range and it is important for agents to stay close to each other while moving in the arena. One example of collective movements is the formation movement, where agents are required to keep a fixed distance and angle relative to other agents within their

neighbourhood. Applications of collective movement include search and rescue, tasks distributed sensing grid, lawn-mowing, vacuum cleaning, box pushing (Kube & Zhang 1992), foraging (Jones & Mataric 2003), *etc.*

In this Chapter, the research is focused on analysing the aggregation behaviour of large groups of agents that follow swarm robotics control paradigms. In particular, we model how a large group of agents would behave in the existence of an artificial attractor while flocking in the arena. This work is inspired by the observation of phototactic organisms, such as moths which fly towards a light source. In the simulations carried out in this chapter the light source is modelled as an Artificial Potential Field (**APF**) to attract agents.

The remainder of this Chapter is organised as follows. In the next section, the simulation approach will be described. The simulation methodologies are then explained and some snapshots of pre-simulation runs are offered. After that the evaluations of each model are shown, and this is followed with discussions.

4.3 Simulation Approach

In this study, a freeware simulation tool called NetLogo (2008) has been used. In NetLogo, the 2-D world is made up of turtles, patches and an observer. Turtles or turtle breeds can be used to define mobile objects. The patches will define the floor or ground in which turtles can move around on. The patches can also be used to define any other visible or invisible objects in the arena. Turtles and patches can have individual variables and characteristics and can follow some set of predefined rules. The observer in the model will be able to oversee everything that is going on in the world.

The model for simulation is based upon its participants, we name them as *agents*, *arena* and the *object*; and sets of rules. The *object*, in our case is a static object which we define as turtle "breed". The rules determine the behaviour of each individual participants, and also specify the way in which these participants will interact with each other.

As mentioned above, the arena has been defined based on patches. In these simulations a spherical or wrapped around working arena size of 201 by 201 patches was chosen. The size is sufficiently large to accommodate the large number of agents that we intend to simulate. At the centre of the arena as shown in the Figure 4.1, patch coordinate of (0, 0), an object called *attractor* has been defined and placed. The *attractor* releases an APF in the arena.

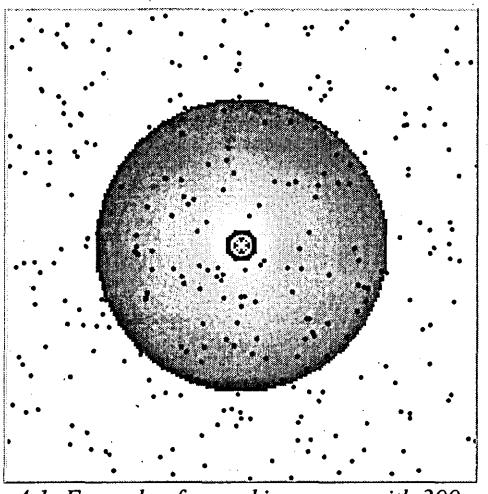


Figure 4.1: Example of a working arena with 300 agents.

Agents are declared as a turtle breed, which are mobile agents. In NetLogo, agents can concurrently carry out some instructions and interact with other agents. Breeds are groups of mobile agents that have same characteristics and follow the same set of rules. The agent has been modelled such that each agent can sense or perceive others around its neighbourhood in 360-degrees within its *visibility range*, as shown in Figure 4.2. *Visibility range* is the variable where we define how far each agent can see or sense from its position; while the *movement span* is a set of maximum angles that are available for the agent to change its direction either to the left or right for its very next movement step.

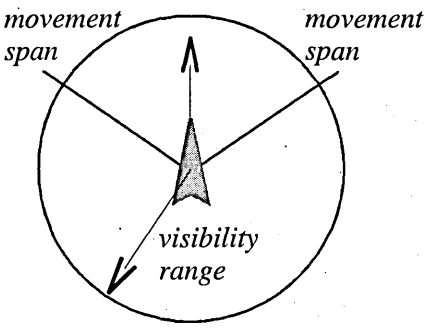


Figure 4.2: Representation of an individual agent

In this Chapter, Wilensky's (1998) flocking model has been used and extended. The model is an attempt to model and mimic the flocking of birds which is inspired by the Reynolds' flocking model (Reynolds 1987). As in Reynolds' model, Wilensky's model does not have any predefined leader and all agents follow the three strategies of flocking, *i.e.* separation, alignment and cohesion. For these strategies, Wilensky limits the turning angle of each strategies using variables called *max-separate-turn*, *max-align-turn* and *max-cohere-turn*. As the names of the variables imply, *max-separate-turn* represents the maximum angle an agent can take during separation strategy; *max-align-turn* is for alignment and *max-cohere-turn* is for the cohesion strategy respectively.

Even though the cohesion and alignment strategies in the Wilensky's model are similar to the Reynolds' model, the separation strategy is slightly different. In Reynolds' model, the separation strategy takes into consideration a number of agents in the neighbourhood of which a distance is maintained. On the other hand, in Wilensky's model, only the closest agent to itself is considered. In this strategy, the agent uses *max-separate-turn* angle and turns away from the closest agent.

4.4 Methodology and Implementation

4.4.1 Simulation methodology

As mentioned in the previous section, for this study, Wilensky's (1998) flocking model has been adopted and adapted. The *attractor* in the centre of the arena releases an APF from its

position as defined by equation (4.1) below.

$$field = \begin{cases} 1 & if\ distance \geq fieldRadius \\ \dfrac{fieldRadius}{distance} & otherwise \end{cases} \quad (4.1)$$

The strength of the APF's *field* is dependent on the patch's distance from its origin, which in our case is represented by the *attractor*. The circular area of the field is subject to a variable, *fieldRadius* which has been set to 63. As shown in Figure 4.1, the white background represents the area which is not affected by the applied field.

The number of agents in the simulations are varied between 100 and 500, with 100 increments. At the beginning of all the simulations, agents are randomly distributed in the arena, which are represented as small black dots as shown in the Figure 4.1.

Within this study, three different movement models have been modelled, namely fish-like, mosquito-like and firefly-like. Note that we are not modelling the movement of fish, mosquito or firefly; the name simply implies the type of observed collective movement of agents in the arena under the different parameter sets. The differences between each movement models are due to the *movement span* and *visibility range* of each agent respectively. As shown in Table 4.1, *visibility range* and *movement span* for the fish-like model have been set to 10 unit-patches and 10-degrees; for the mosquito-like model 7 unit-patches and 45-degrees, while for the firefly-like model they are 5 unit-patches and 90-degrees, respectively.

Table 4.1: Variables for movement models

movement model	*movement span*	*visibility range*
Fish-like	10	10
Mosquito-like	45	7
Firefly-like	90	5

As Wilensky's model of flocking (1998) is being extended, three more variables from the original model needed to be introduced; *max-align-turn*, *max-cohere-turn* and *max-separate-turn*. These variables are the maximum angles that each agent can turn through during the *alignment*, *cohesion* and *separation* rules respectively.

For these simulations, those three angles rely on the *movement span* angle; which is the maximum turning angle of each agent for its next movement or time step. As shown in Table 4.2, the value for *max-align-turn* is set to half of the *movement span* angle, and *max-cohere-turn* and *max-separate-turn* to one-third of the *movement span* respectively. These values have been chosen based on our observations during the pre-simulations run such that each movement model exhibit "realistic" flocking. In "realistic" flocking agents are free to leave and enter the flock, just as biological organism do.

Table 4.2: Flocking variables for each movement model

movement model	*movement span*	*max-align-turn*	*max-cohere-turn*	*max-separate-turn*
Fish-like	10	10/2	10/3	10/3
Mosquito-like	45	45/2	45/3	45/3
Firefly-like	90	90/2	90/3	10/3

Throughout this Chapter, the agent's velocity is fixed to one unit displacement, whilst the agent's heading H, varies over time. The separation between agents (*minimum separation*) has been set to two units of displacement, which seems a reasonable figure considering that the velocity is one unit of displacement for each time step. The change of heading (H) is subject to the APF and flocking rules which consist of separation, alignment and cohesion strategy as previously mentioned. In the separation strategy, the agent only considers the nearest agent, the heading for separation ($H_{separation}$) is defined as follows:

$$H_{separation} = \begin{cases} H_{current} + max-separate-turn & if\ H_{current} - H_{nearest\ neighbour} \geq 0 \\ H_{current} - max-separate-turn & otherwise \end{cases} \quad (4.2)$$

where $H_{current}$ is the agent's current heading and $H_{nearest\ neighbour}$ is the nearest neighbour's current heading. In this strategy the agent will turn away from its nearest agent by *max-separate-turn*.

In the alignment strategy, the agent will change its heading with a similar heading or the average heading of its neighbours. The heading for alignment ($H_{alignment}$) is defined as follows:

$$H_{alignment} = \begin{cases} H_{align} & if\ |H_{current} - H_{align}| \leq max-align-turn \\ H_{current} + max-align-turn & else\ if\ H_{current} < H_{align} \\ H_{current} - max-align-turn & else \end{cases} \quad (4.3)$$

where H_{align} is the average heading of neighbour(s) within neighbourhood area. H_{align} is defined as the following equation:

$$H_{align} = \frac{1}{n_a} \sum_{neighbour=1}^{n_a} H_{neighbour} \quad (4.4)$$

where n_a is the number of neighbours within the *visibility range* and $H_{neighbour}$ is the heading of a particular neighbour.

In the cohesion strategy, the agents will try to stay close to its neighbours. The heading for cohesion ($H_{cohesion}$) is defined as:

$$H_{cohesion} = \begin{cases} H_{cohere} & if\ |H_{current} - H_{cohere}| \leq max-cohere-turn \\ H_{current} + max-cohere-turn & else\ if\ H_{current} < H_{cohere} \\ H_{current} - max-cohere-turn & else \end{cases} \quad (4.5)$$

where H_{cohere} is the heading towards the centroid of agents in the neighbourhood and defined as:

$$H_{cohere} = H\left(\frac{1}{n_a} \sum_{neighbour=1}^{n_a} x_{neighbour}, \frac{1}{n_a} \sum_{neighbour=1}^{n_a} y_{neighbour}\right) \quad (4.6)$$

where n_a denotes the number of agents within neighbourhood range; $x_{neighbour}$ and $y_{neighbour}$ are the neighbour's x-coordinate and y-coordinate respectively; and $H(x, y)$ means set the heading towards the coordinate of (x, y). As previously mentioned, we have set the maximum turning angle for each strategy as in Table 4.2. If the computed turning angle ($|H_{current} - H_{cohere}|$ or $|H_{current} - H_{align}|$) is larger than the turning limit (*max-cohere-turn*, *max-align-turn*), then the maximum turning angle will be used, such shown in the equations (4.3) and (4.5).

The movement models in this Chapter are governed by rules as represented in the flowchart in

Figure 4.3. From the flowchart, it is clear that in the movement models we have four different phases or four behaviours which are represented by the rectangular boxes, *i.e. wander*, *wander inside field*, *flock* and *flock inside field*. In the *wander* state, the heading of each agent (H_{wonder}) is determined by the following way:

$$H_{wander} = H_{current} + random(movement\ span) \tag{4.7}$$

where $H_{current}$ is the current heading; and "random (*movement span*)" generates a random number which is between *-movement span* and *+movement span*. During the *wander* phase, agents randomly change their heading either to the right or left depending on the positive or negative sign of the generated *movement span*. If it is positive, it will turn to the right whilst negative for the left.

For the *flock* behaviour, agents will first compute the distance of their nearest neighbour. The distance is then compared with a variable called *minimum separation*. If the computed distance is smaller or equal to the *minimum separation*, agents will use $H_{separate}$ (eq. 4.2) as the next heading. The heading of the agent during *flock* phase (H_{flock}) is decided in the following manner:

$$H_{flock} = \begin{cases} H_{separation} & if\ distance_{nearest\ neighbour} \leq minimum\ separation \\ H_{alignmentCohesion} & otherwise \end{cases} \tag{4.8}$$

where $H_{alignmentCohesion}$ is the average heading of $H_{alignment}$ and $H_{cohesion}$ and given as follow:

$$H_{alignmentCohesion} = \frac{H_{alignment} + H_{cohesion}}{2} \tag{4.9}$$

For the primitive heading inside the field of APF (H_{field}) which represents as the attraction towards and repulsion against the centre of the APF has been computed in the following way

$$H_{field} = \begin{cases} H_{attractor} \mp \left(90 + \frac{field}{2}\right)^{o} & if\ \ field > 10 \\ H_{attractor} \mp \left(90 - \frac{10}{field}\right)^{o} & otherwise \end{cases} \tag{4.10}$$

where *field* is the strength of the APF (as in eq. 4.1), $H_{attractor}$ is the heading towards the centre of APF or *attractor*. Whenever the agent is close to the *attractor* or the *field* of the patch that the

agent is at is greater than 10, the agent will repulse or turn away from the *attractor.* Beyond that, the agent will be attracted towards the *attractor.* In the above equation (4.10), there are two operators in the equations, these operators are dependent on the which side the *attractor* is at relative to the agent in question. If the *attractor* is on the right side of the agent, the top operator will be used; if the attractor is on the left, the bottom operator will be used otherwise.

For *wander inside field* and *flock inside field* behaviours, the heading of agents are determined by averaging the headings of the respected strategies, with the primitive heading inside the *field* (H_{field}). The headings for these behaviours are defined as the following equation:

$$H_{wanderField} = \begin{cases} \frac{H_{wander}+H_{field}}{2} & if \ \frac{H_{wander}+H_{field}}{2} < movement\ span \\ H_{current} + movement\ span & else\ if \ H_{current} < \frac{H_{wander}+H_{field}}{2} \\ H_{current} - movement\ span & otherwise \end{cases} \quad (4.11)$$

$$H_{flockField} = \begin{cases} \frac{H_{flock}+H_{field}}{2} & if \ \frac{H_{flock}+H_{field}}{2} < movement\ span \\ H_{current} + movement\ span & else\ if \ H_{current} < \frac{H_{flock}+H_{field}}{2} \\ H_{current} - movement\ span & otherwise \end{cases} \quad (4.12)$$

where $H_{wanderField}$ is the agent's heading while in the *wander inside field* state; and $H_{flockField}$ is the heading for *flock inside field.*

As the simulation starts, each agent enters either the *wander* or *wander inside field* state depending on the agent's current position as shown in the flowchart in Figure 4.3. If the agent's position is not affected by the APF, it will fall into the *wander* phase, otherwise it will be the *wander inside field.* While in the *wander* or *wander inside field* phase, at each simulation time step agents will have a chance to change its heading randomly, but within the constraints of the *movement span* limit. Each agent then examines the position where they were at; if that particular position is affected by the APF, or having a *field* value of larger than one (*field* > 1),

the agent is then attracted to the centre of the field.

The agent then looks around, within its vicinity or *visibility range,* for flockmates. If any mate is found, the agent flocks with the flockmates, otherwise it continues roaming. While inside the *field*, the aforementioned rules were used with added attraction to the centre of the *field*, so that agents will not leave the *field*.

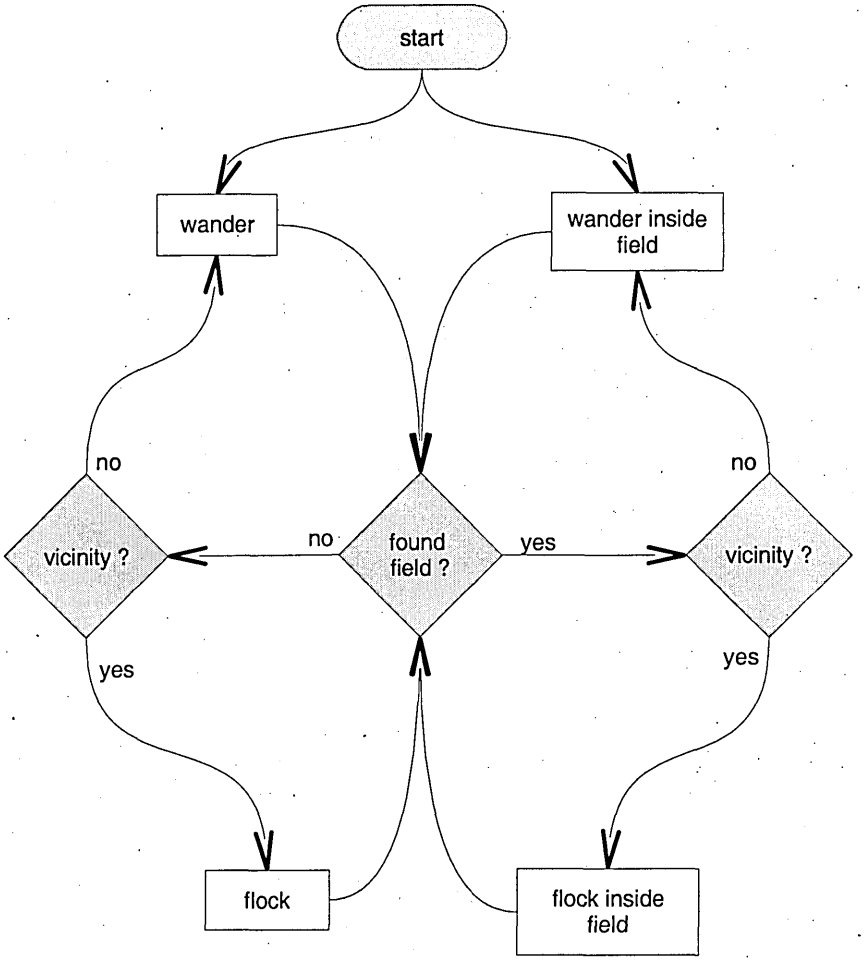


Figure 4.3: Flowchart of movement models

Figure 4.4 and Figure 4.5 show some sample trajectories and turning angle plots for each model after we apply each different movement span respectively. From Figure 4.4 and Figure 4.5 we can clearly see the differences between the trajectories and the plot of turning angle against time of each movement model. Figure 4.4(a) shows fish-like motion where the movement is like fish motion with a "calm" turning angle. Fish-like motion is useful for scanning large areas of the arena in a short time period. Figure 4.4(b) and (c) show the trajectories of the mosquito-

like and the firefly-like movement model, respectively. As we can see from the trajectories, the firefly-like motion allows the agent to move around scanning in a small local area, and this can be useful for searching for small objects in a small area, while the mosquito-like movement appears to scan a wider area in the arena as well as its own neighborhood area.

4.4.2 Pre-simulation runs

In the pre-simulation runs, each movement model was assessed without the *attractor* which releases the APF to see how the the agents would behave in the arena. We started the simulations with 200 agents randomly distributed in the arena and allowed the simulations to execute for 1,000 time steps.

Figure 4.6(a), (b) and (c) show the aggregation of fish-like, mosquito-like and firefly-like swarms movement models, respectively. From Figure 4.6(a) for the fish-like movement model, the aggregation pattern that emerges shows that the agents congregate in large numbers in several groups. The firefly-like movement model (Figure 4.6(c)), on the other hand, shows that agents formed several clusters with a smaller number of agents in each cluster.

In the mosquito-like movement model, Figure 4.6(b), the agents aggregate in several large and small groups. This behaviour is similar to what Ikawa and Okabe (1997) suggested, that mosquitoes do not remain at a single swarming site but repeatedly enter and leave the sites. For this reason, in nature mosquitoes aggregate with large and small numbers in each group; hence, the name mosquito-like movement model.

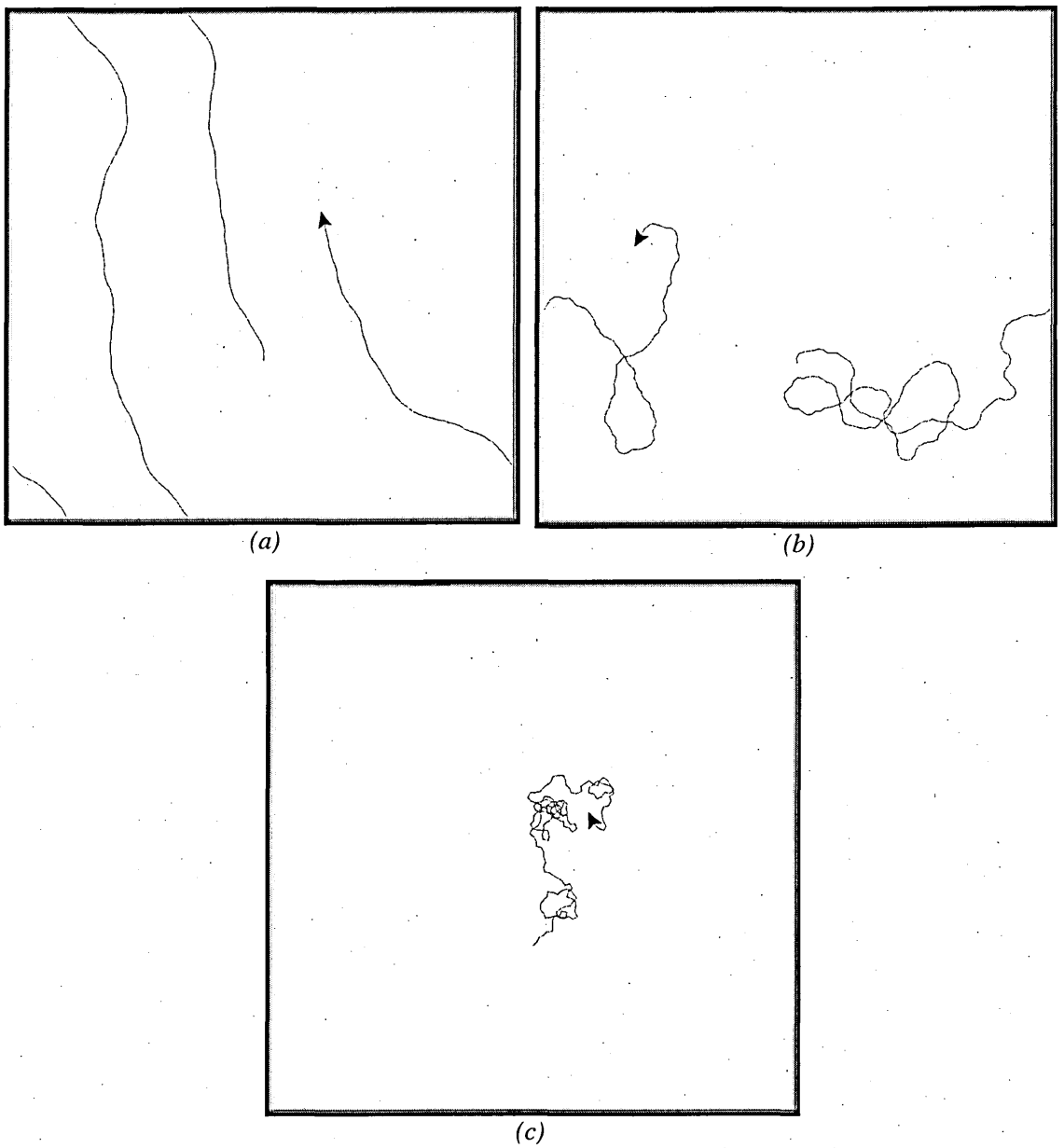
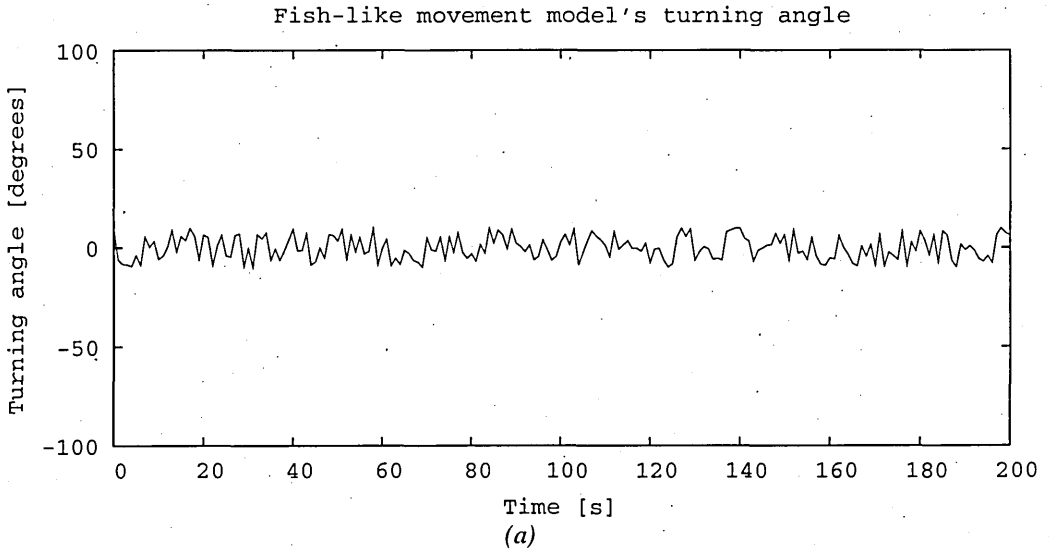
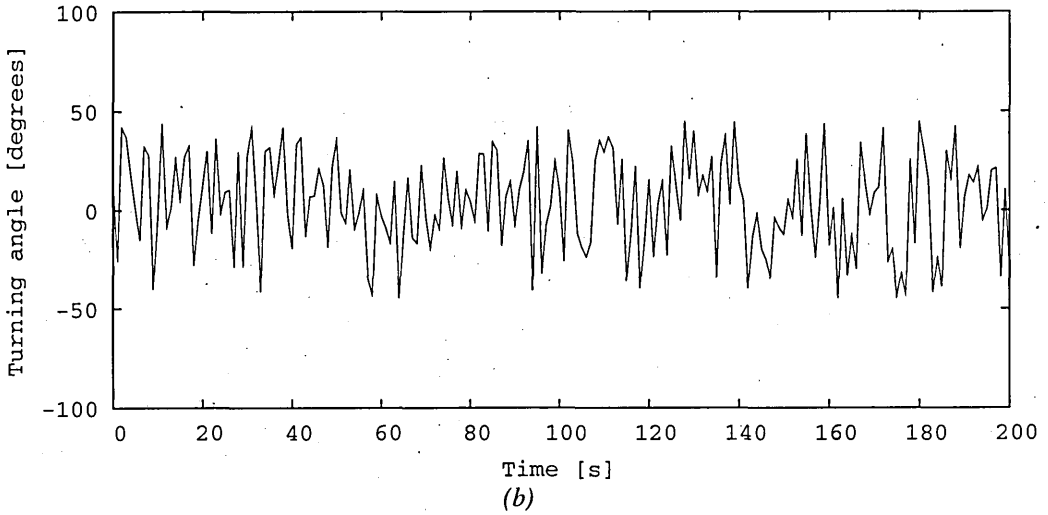


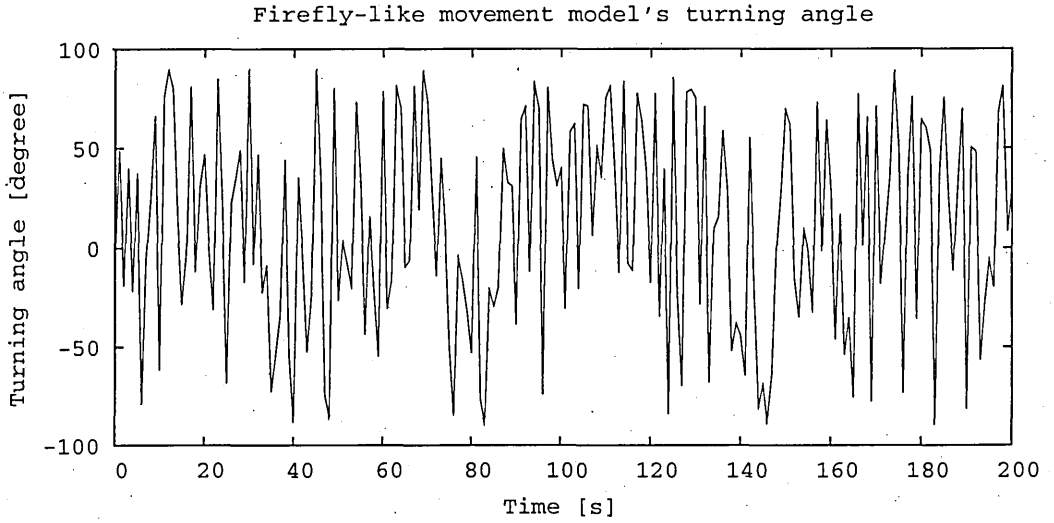
Figure 4.4: Agents motion trajectories for each movement model: (a) fish-like, (b) mosquito-like, (c) firefly-like



(a)



(b)



(c)

Figure 4.5: Agents motion trajectories for each movement model: (a) fish-like, (b) mosquito-like, (c) firefly-like

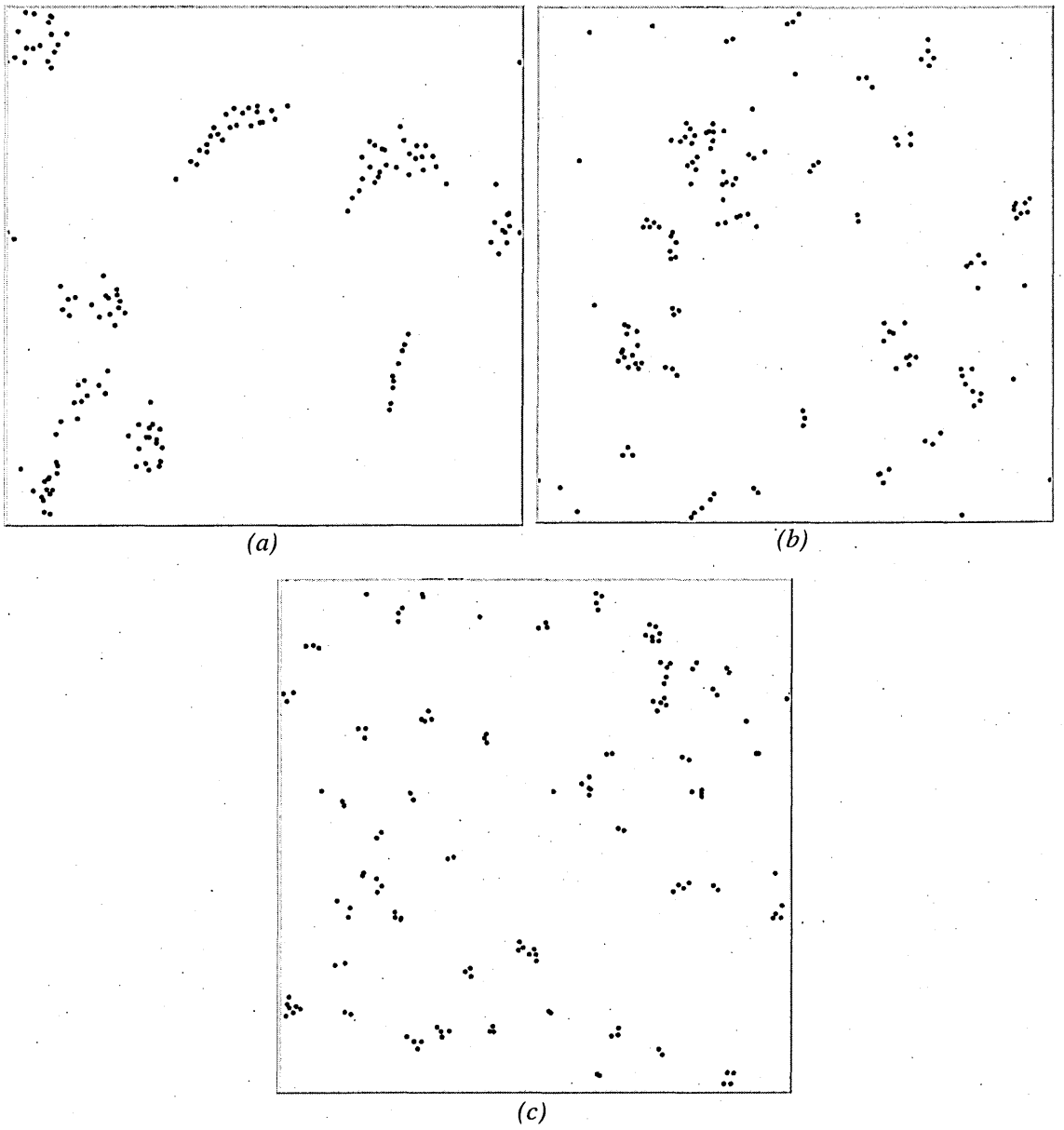


Figure 4.6: Agents position at t = 1000 time steps of three movement models: (a) fish-like, (b) mosquito-like, (c) firefly-like.

4.5 Evaluation

As stated previously, several different numbers of agents were used in these simulations; *i.e.* 100, 200, 300, 400, and 500 number of agents were used. All simulations used a torus-wrapped square arena of size 201 by 201 patches, such as the one shown in Figure 4.1. Thirty runs are made for each movement model and each different number of agents with random initial placement of the agents in the arena. The *fieldRadius* for the APF has been set to 63 (page 102). This is to give sufficient space for all 500 agents to reside in the APF's field considering the

minimum separation of 2 units of displacement (page 103) between agent.

The performance was evaluated at the end of the simulation and all runs were executed for 7,000 simulation time steps to provide enough time for all agents to aggregate towards the *attractor* which releases APF. The data for analysis was recorded at every 200 time steps during these simulations.

4.5.1 Evaluating the fish-like movement model

In evaluating each movement model, the number of agents within the circular area of the *attractor* is first counted, or the circular area starting from the centre of the field, in our case, from the patch at (0,0). As the number of agents in the simulations was fixed (varies from 100 to 500 with increment of 100 agents), and the working arena at 201 by 201 patches, increases from zero, we can expect that the number of agents should reach a maximum number when the radius of the circular area originating from the the centre of *field* reaches 141, as it would completely cover the arena. The reason for counting the number of agents within the circular area was to pre-determine how close these agents are to the *attractor.*

As mentioned previously, for the fish-like movement model, the *movement span* is set to 10-degrees and *visibility range* to 10 patches. Figure 4.7 shows the agent's location from one of the simulations with 300 agents at three different simulation time steps of 150, 330 and 500 time steps, respectively. From the figure, it is clear that as early as 150 time steps, more than half the number of agents have already converged towards the centre of the arena or towards the *attractor.*

During the *flock inside field* phase, the flocking agents exhibited a smooth circling behaviour concentrated on the origin of the APF; in this case, the centre of the arena or the *attractor.* The overall direction of the flow appears to be random, sometimes clockwise and sometimes anti-clockwise. The reason for this is because as soon as an agent enters the *field* it will search around for flockmates. If any is found, it will change its direction to match the majority of its flockmates in either a clockwise or anti-clockwise direction, resulting in the aforementioned

emergent behaviour inside the *field.*

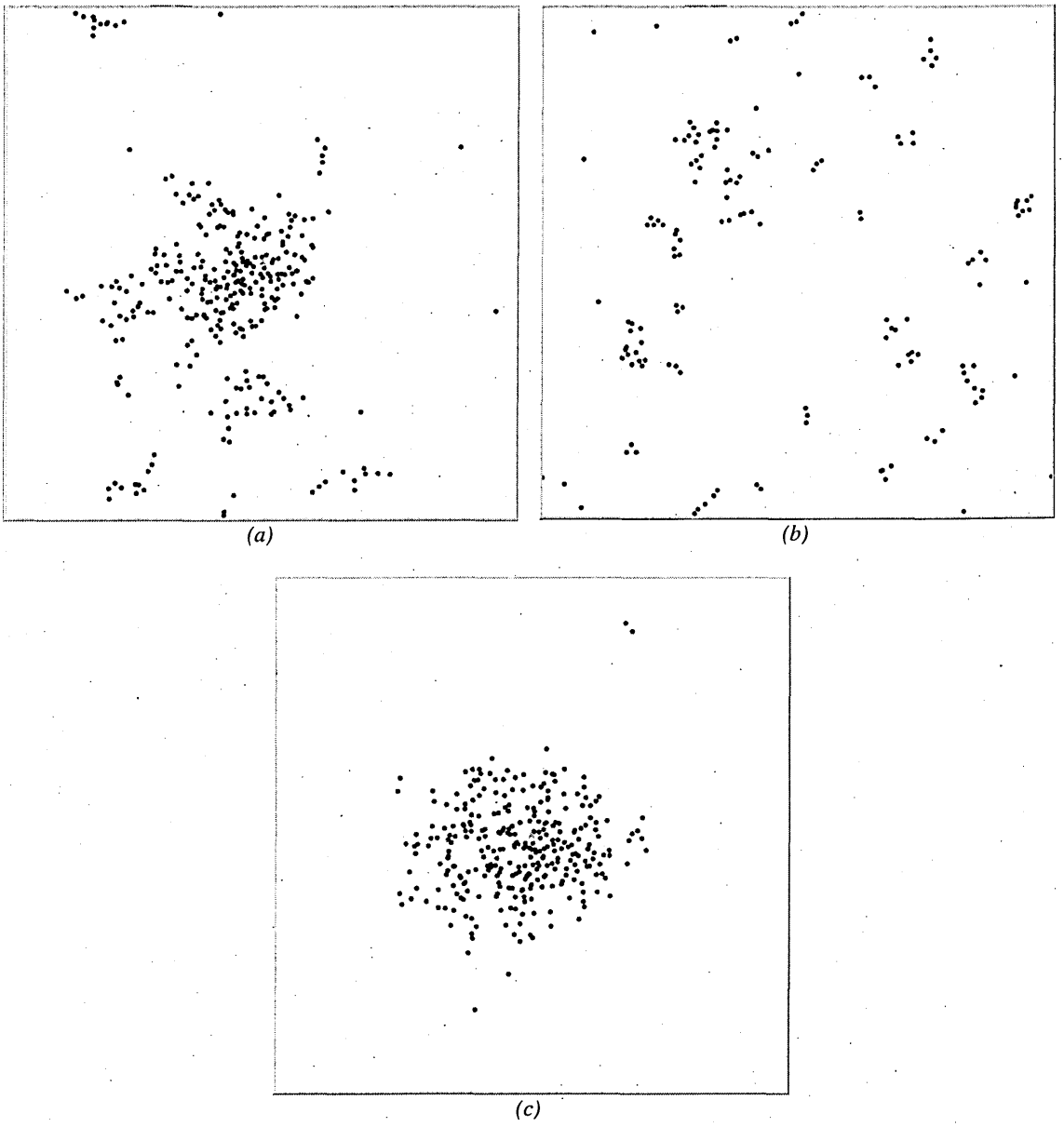


Figure 4.7: Positions of 300 agents in the arena at different time steps for the fish-like movement model from one of the simulation runs; (a) at t=150, (b) at t=330, (c) at t=500 time steps.

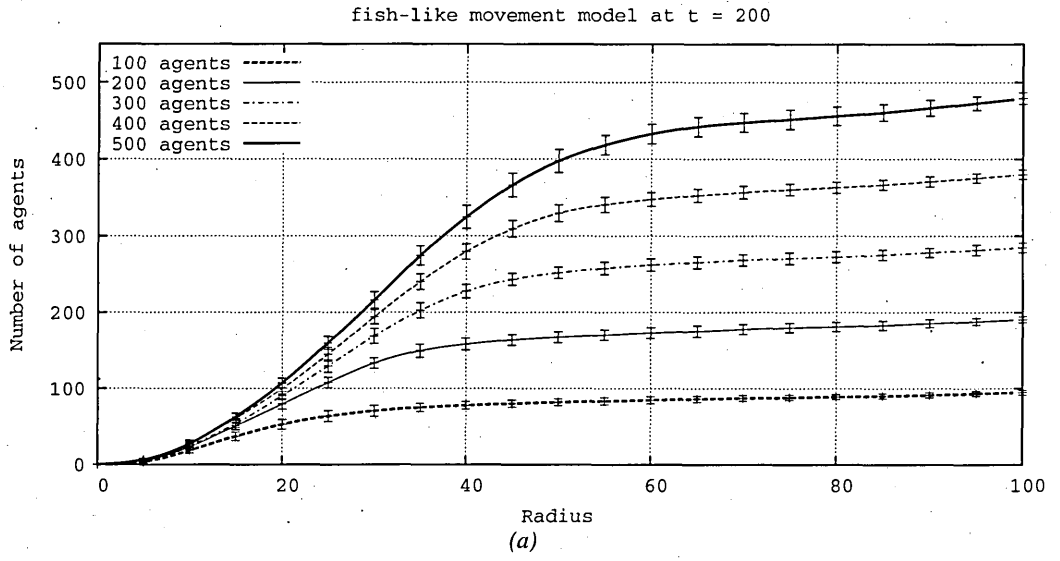
Table 4.3 provides a summary of the number of agents within the 60-patch radius from the centre of the APF; Figure 4.8(a) and (b) are the plots of the number of agents within the circular area from the centre of the APF for the fish-like movement model, at simulation time steps of 200 and 600, respectively. The results show that, at t = 200 simulation time step, about 85% of the agents that are participating in the simulations are already inside the field; in other words, about 85% of the agents in the particular simulations have already converged towards the

attractor, with a standard deviation of less than 5%. The highest standard deviation is observed in the simulation consisting of 100 agents; this can be explained by the fact that the lesser the number of agents in the arena, the more time the agents need to scan through arena. At $t = 600$ simulation time step, almost all the agents in the arena are already aggregated near the *attractor* with standard deviations of 1.5% for 100 agents, and less than 1% for 200 and more agents.

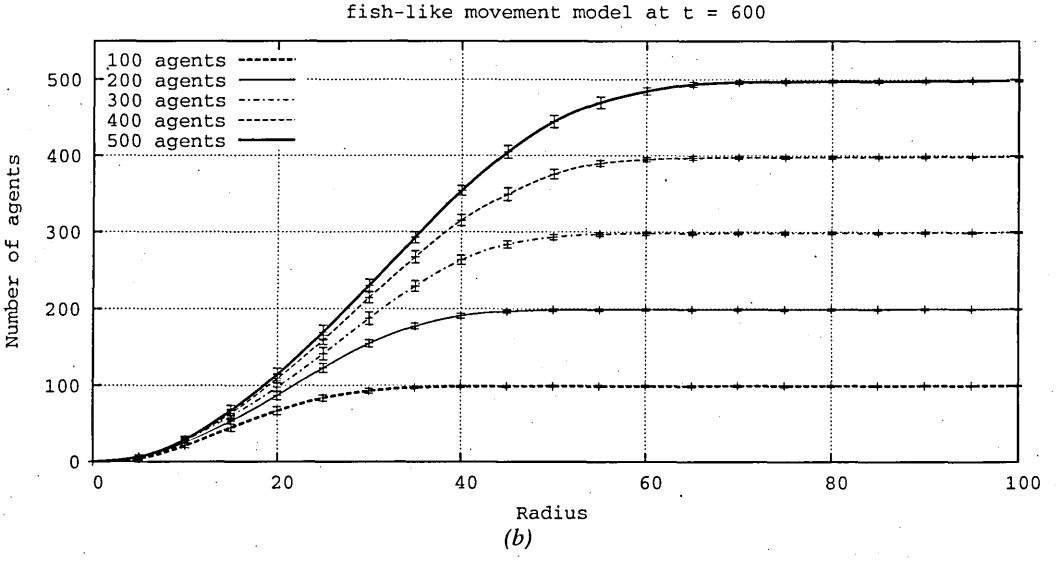
Table 4.3: Fish-like movement model

Time	$t = 200$					$t = 600$				
Total number of agents	100	200	300	400	500	100	200	300	400	500
Mean number of agents within 60-patch radius from the centre of APF	84.7	173.2	262.1	347.7	432.8	99.0	198.9	297.9	394.7	484.5
% of agents within 60-patch radius from centre of APF	84.7	86.6	87.4	86.9	86.6	99.0	99.5	99.3	98.7	96.9
standard deviation	4.6	6.9	7.9	8.8	12.7	1.5	1.6	2.4	3.0	4.7
% of standard deviation	4.6	3.4	2.6	2.2	2.5	1.5	0.8	0.8	0.7	0.9

Figure 4.9 shows the simulation plots with 100 agents; the number of agents within the circular area from the *attractor* at 200, 400 and 600 simulation time step respectively. From the plots, it can be clearly seen that the curves differ. At $t = 200$, the number of agents increases gradually with noticeably large standard deviation; while at $t = 400$ and $t = 600$, the standard deviations decrease, showing that the agents movement have stabilised.



(a)



(b)

Figure 4.8: Number of agents for the fish-like movement model within circular area from the attractor; (a) at t = 200, (b) at t = 600 simulation time steps.

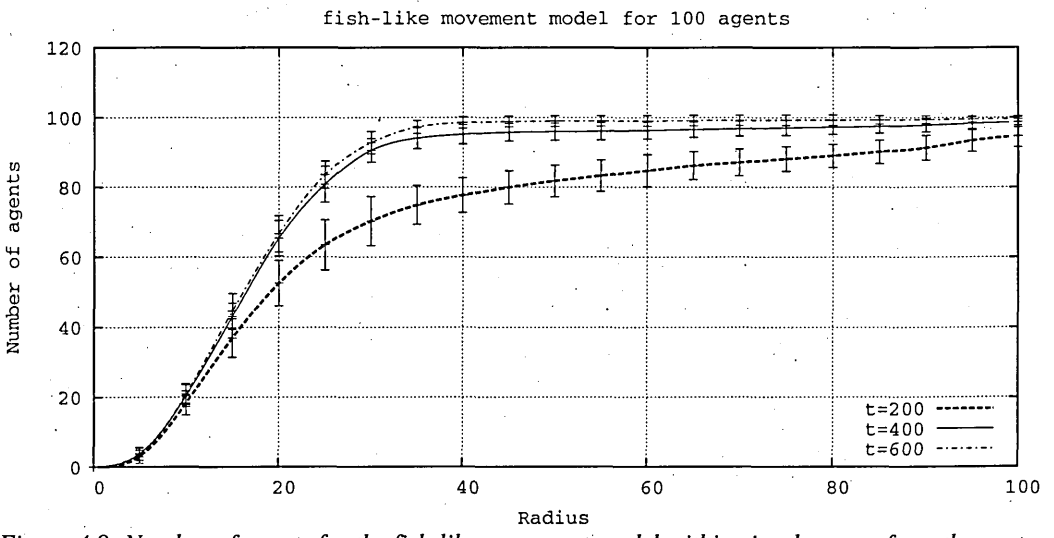


Figure 4.9: Number of agents for the fish-like movement model within circular area from the centre of APF at different simulation time steps.

4.5.2 Evaluating the mosquito-like movement model

As mentioned previously, for the mosquito-like movement model, *visibility range* and *movement span* have been set to 7 patches and 45-degrees, respectively. In this movement model, without the attractor in the arena, agents appear to be form several clusters of varying sizes as shown in Figure 4.6(b).

Figure 4.10 shows the snapshots of one of the simulation runs for the mosquito-like movement model with 300 agents at three different time steps: 500, 1000 and 1500 time steps respectively. At t = 500, we notice that more than two-third of the agents have already converged towards the centre of arena, at t = 1000, the number of agents is increasing, and at t = 1500 almost all the agents have found the APF releases by the *attractor*, resulting the agents aggregate near the centre of the arena.

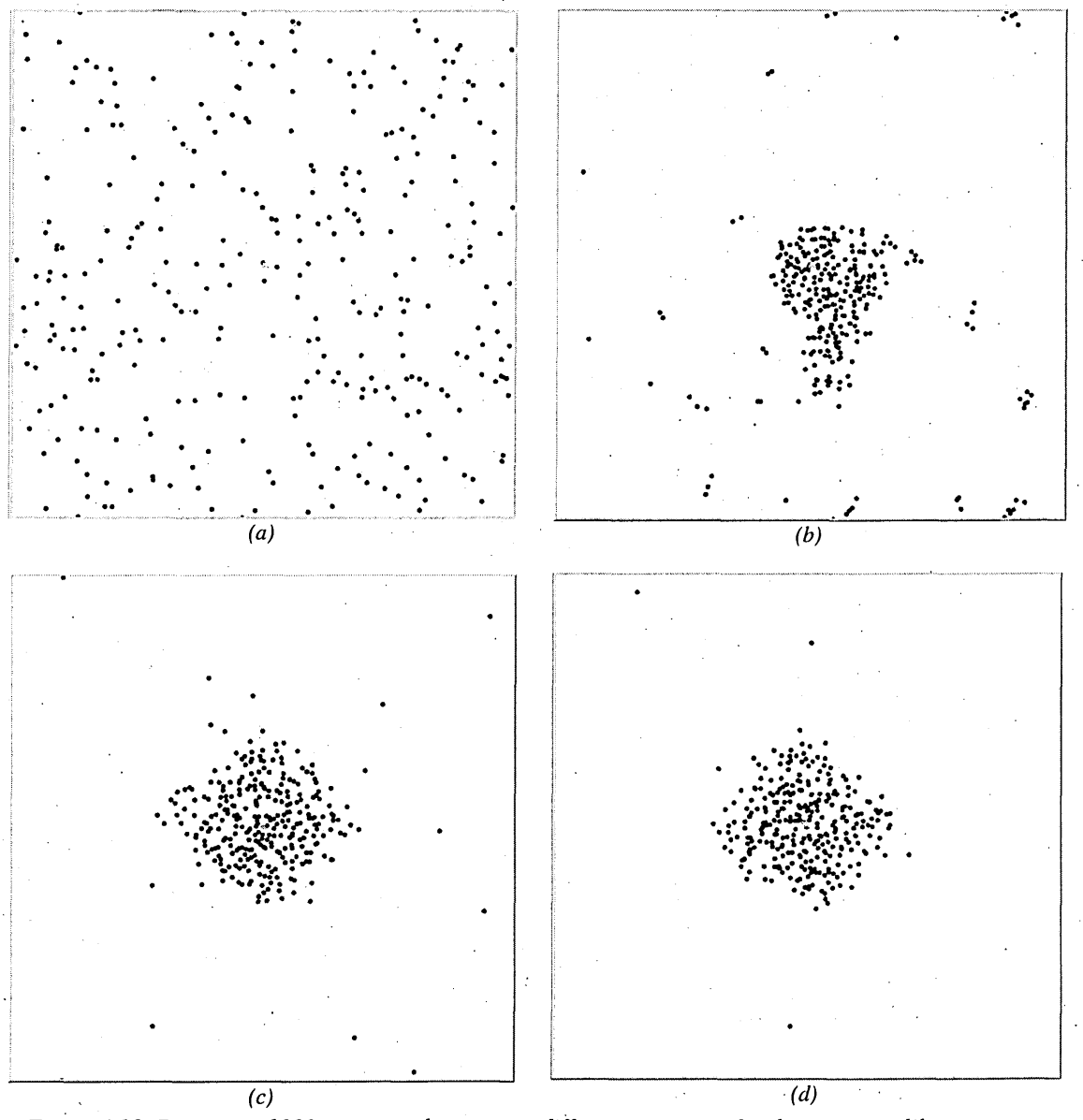


Figure 4.10: Positions of 300 agents in the arena at different time steps for the mosquito-like movement model; (a) at t=0, (b) at t=500, (c) at t=1000, (d) at t=1500 time steps.

During the *flock inside field* phase, the agents appear to move in a circulating motion around the origin of the APF, but not as smoothly as that exhibited by the fish-like movement model. In this case the agents tend to stay a little closer to their flockmates, thus limiting the circulating movement. The emerged motion is brought about by the need for the agents to move, but the direction, whether clockwise or counterclockwise is indeterminate.

Table 4.4, Figure 4.11(a) and (b) are the selected data and plots of the results for the mosquito-like movement model's simulations. Table 4.4 shows the number of agents within the 60-patch radius from the *attractor* at the simulation time steps of 400 and 1200.

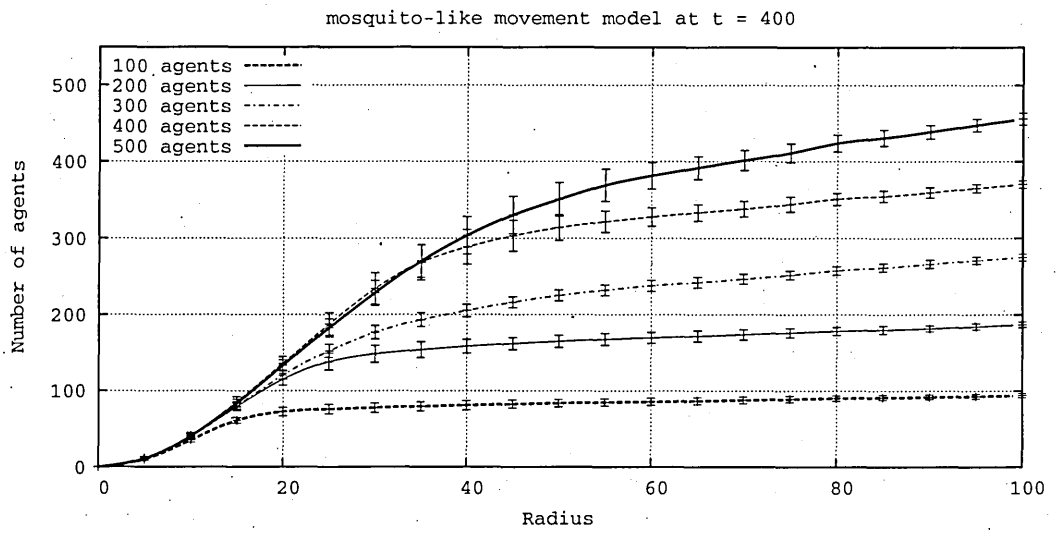
Table 4.4: Mosquito-like movement model

Time	t = 400					t = 1200				
Total number of agents	100	200	300	400	500	100	200	300	400	500
Mean number of agents within 60-patch radius from the centre of APF	85.5	169.3	237.8	327.6	381.4	98.2	196.5	294.1	394.3	488.1
% of agents within 60-patch radius from centre of APF	85.5	84.7	79.3	81.9	76.3	98.2	98.2	98.0	98.6	97.6
standard deviation	4.8	7.1	7.2	12.0	17.4	1.6	3.7	2.1	3.1	3.9
% of standard deviation	4.8	3.6	2.4	3.0	3.5	1.6	1.8	0.7	0.8	0.8

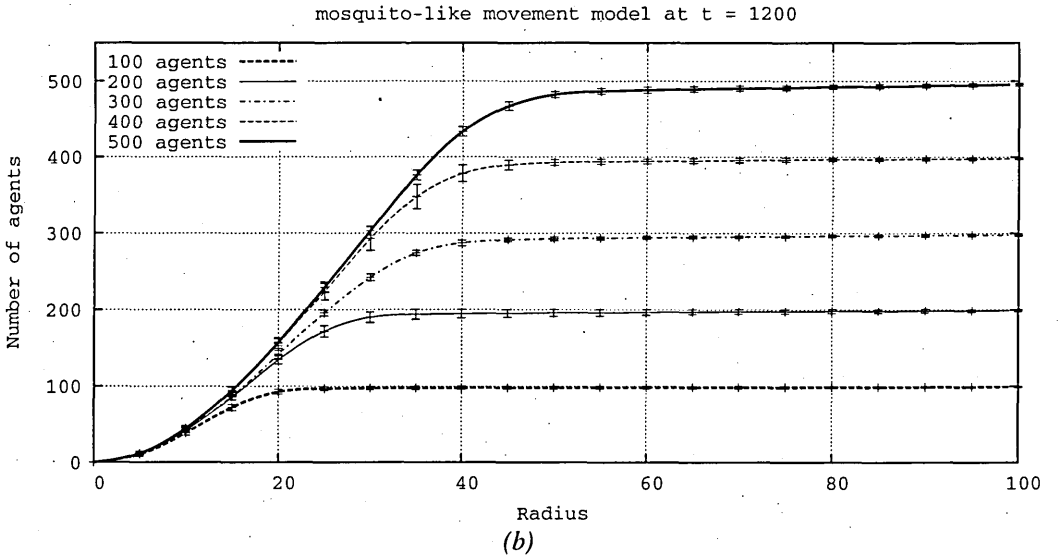
From the results, at t = 400 simulation time steps, about 80% of the agents are within 60 patch radius from the *attractor*. For simulations with 100 agents, there are 85.5% of the agents converged toward the attractor compare to only 76.5% with 500 agents; with standard deviation of 4.8% and 3.5% respectively.

At t = 1200 simulation time steps on the other hand showed that almost all the agents in the simulations are within 60 patch radius from the *attractor*, or inside the APF's field which was released by the *attractor*. At this time the standard deviations are rather small with all of it being less than 2%, and less than 1% for 300 and more agents.

Figure 4.12 is the plot for simulations with 300 agents; the number of agents within circular area from the *attractor* at 400, 600 and 1200 simulation time step respectively. From the plot, it can be clearly seen that the curve differs at each different time steps. As the time increases from 200 to 400, and 600, the number of agents within the APF's field increased accordingly; and the standard deviations show to decrease.



(a)



(b)

Figure 4.11: Number of agents for the mosquito-like movement model within circular area from the attractor; (a) at t = 400, (b) at t = 1200 simulation time steps.

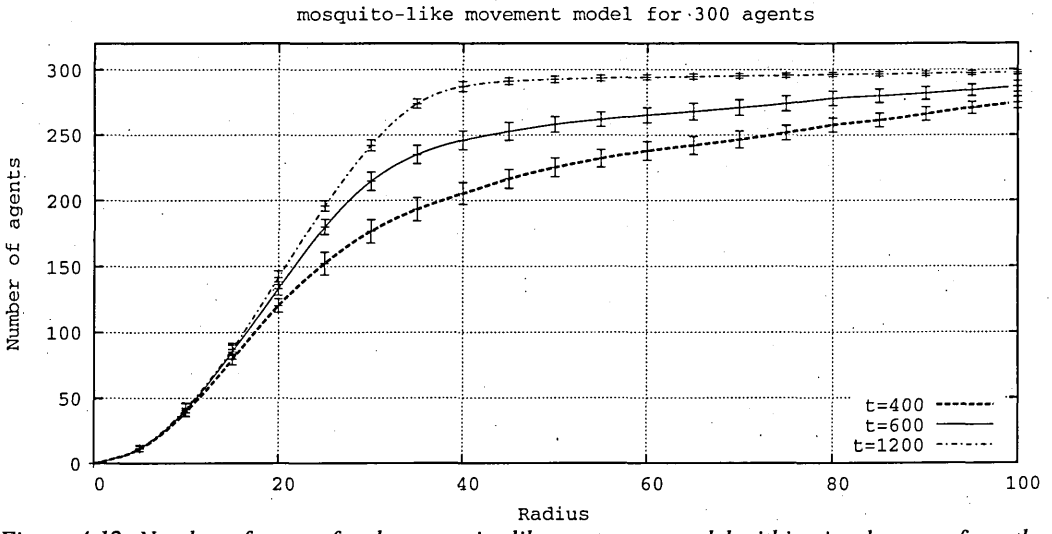


Figure 4.12: Number of agents for the mosquito-like movement model within circular area from the centre of APF at several different simulation time steps.

4.5.3 Evaluating the firefly-like movement model

For the firefly-like movement, the *visibility range* and *movement span* are set to 5 patches and 90 degrees respectively. Figure 4.13 shows one of the simulation runs snapshots for the firefly-like movement model at t = 1000, 2000 and 3000 time steps, respectively. At t = 1000, even though some of the agents have already converged towards the *attractor*, we can clearly see that a great number of agents are still in the *wander* or *flock* phase; in other words, agents are roaming in the arena looking for flockmates or flocking outside the APF's *field.* At t = 2000, the number of agents outside the *field* seems to decrease significantly compared to t = 1000. At t = 3000, almost all the agents are in the *wander inside field* phase or have already converged towards the *field.*

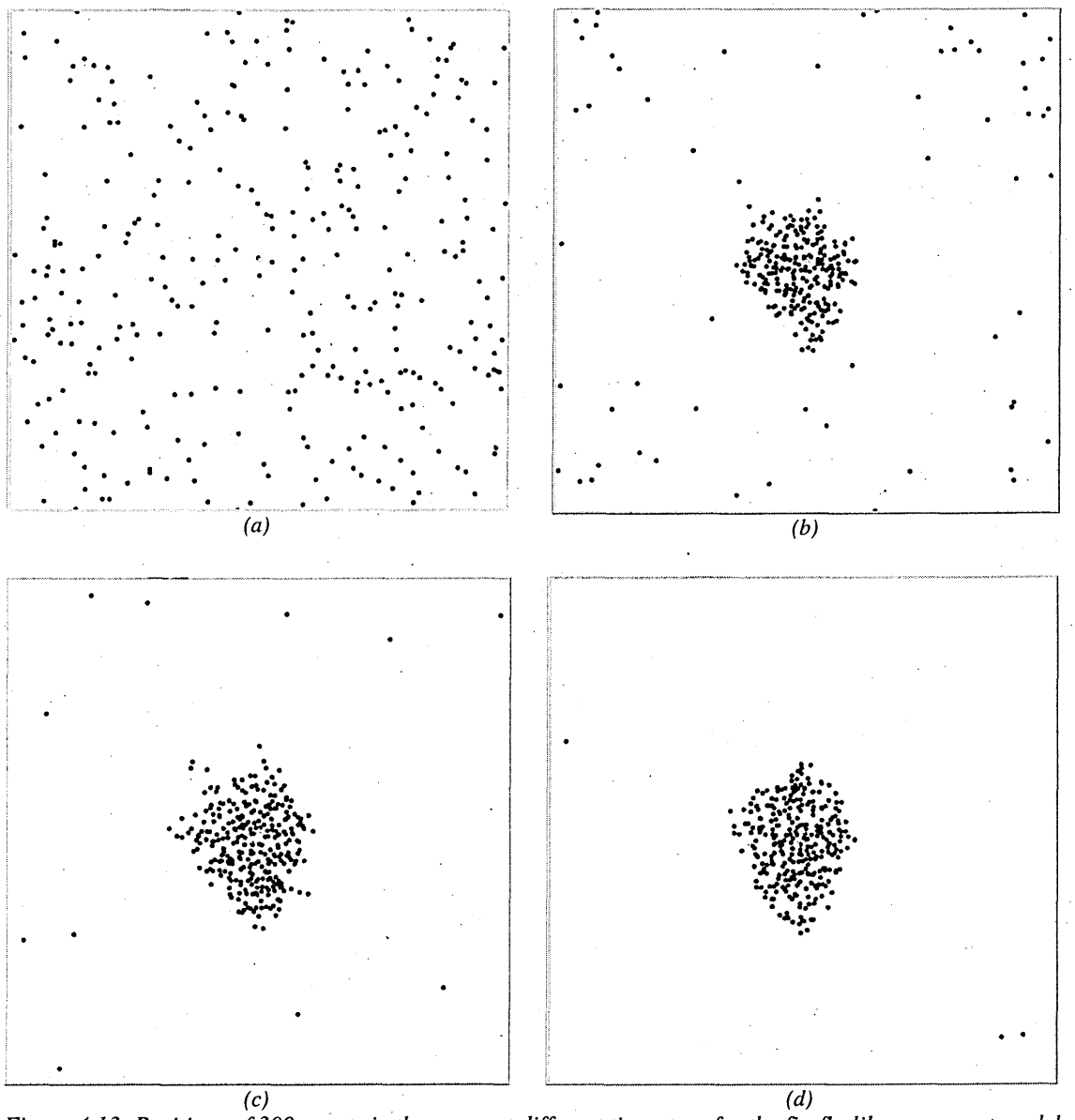


Figure 4.13: Positions of 300 agents in the arena at different time steps for the firefly-like movement model at: (a) t=0, (b) t=1000, (c) t=2000, (d) t=3000 simulation time steps.

During the *flock inside field* phase, unlike the previous two movement models, instead of agents circulating the origin of the APF, the agents seem to only converge to the centre of the APF's *field* and move around only within their small local area.

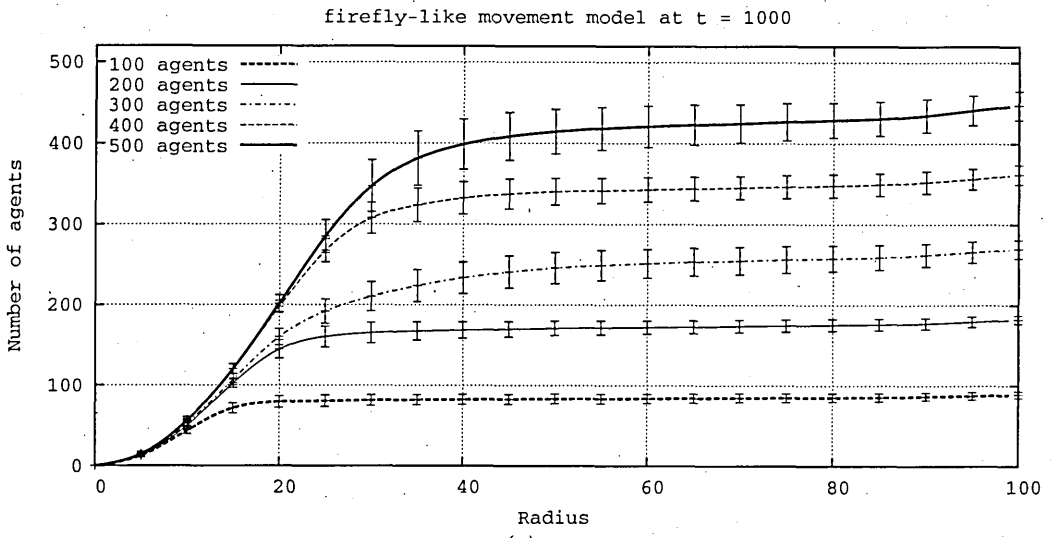
Table 4.5 shows the number of agents within 60-patch radius from the *attractor*; Figure 4.14(a) and (b) are the plots of the number of agents within the circular area from the *attractor* for the firefly-like movement model, at simulation time steps of 1000 and 3000, respectively. Results show that at t = 1000 time steps, about 85% of the agents are already inside the APF's field which has been set to 63 patch radius; in other words, about 85% of the agents have already

converged near the *attractor*, with standard deviations between 3.7% (for 400 agents) and 5.9% (for 300 agents). At t = 3000 simulation time steps, nearly all the agents in the arena already aggregated near the *attractor* with standard deviations of less than 2%.

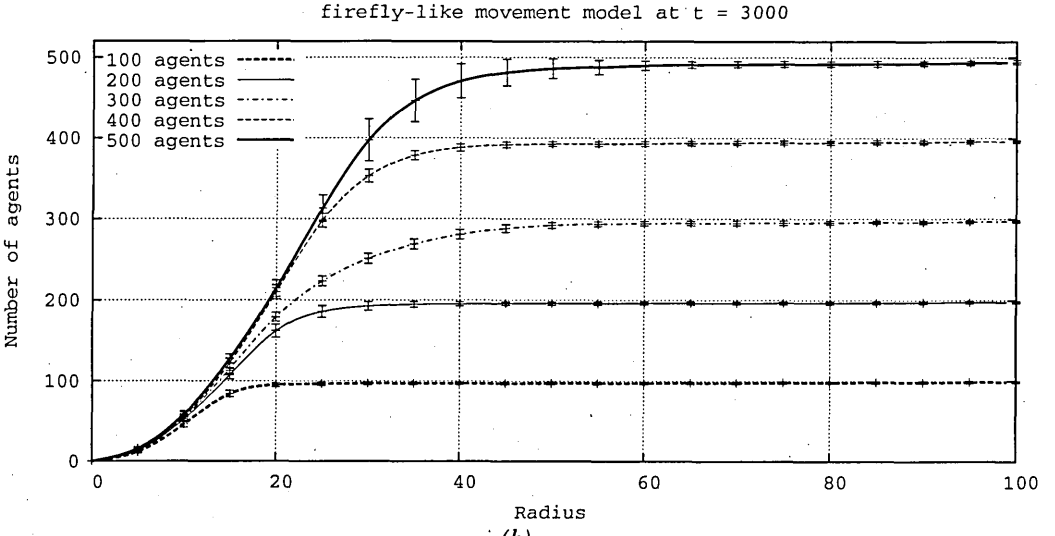
Table 4.5: Firefly-like movement model

Time	t = 1000					t = 3000				
Total number of agents	100	200	300	400	500	100	200	300	400	500
Mean number of agents within 60-patch radius from the centre of APF	83.8	172.4	251.7	343.0	421.0	97.3	196.4	294.5	393.1	490.0
% of agents within 60-patch radius from centre of APF	83.8	86.2	83.9	85.8	84.2	97.3	98.2	98.2	98.3	98.0
Standard deviation	5.8	8.1	17.8	14.9	25.3	1.6	2.0	2.8	2.9	5.7
% of standard deviation	5.8	4.1	5.9	3.7	5.1	1.6	1.0	0.9	1.0	1.1

Figure 4.15 shows the plot for simulations with 300 agents; the number of agents within the circular area from the *attractor* at three different time steps of 600, 2000 and 5000. From the plot it can be clearly seen that at t = 600, more than two-third of agents are already converged toward the *attractor*, or within 60 patch radius from the *attractor*. At t = 2000 and t = 5000, the number of agents within the APF's field increases, and the standard deviations decreases accordingly.



(a)



(b)

Figure 4.14: Number of agents for the firefly-like movement model within circular area from the attractor; (a) at t = 1000, (b) at t = 3000 simulation time steps.

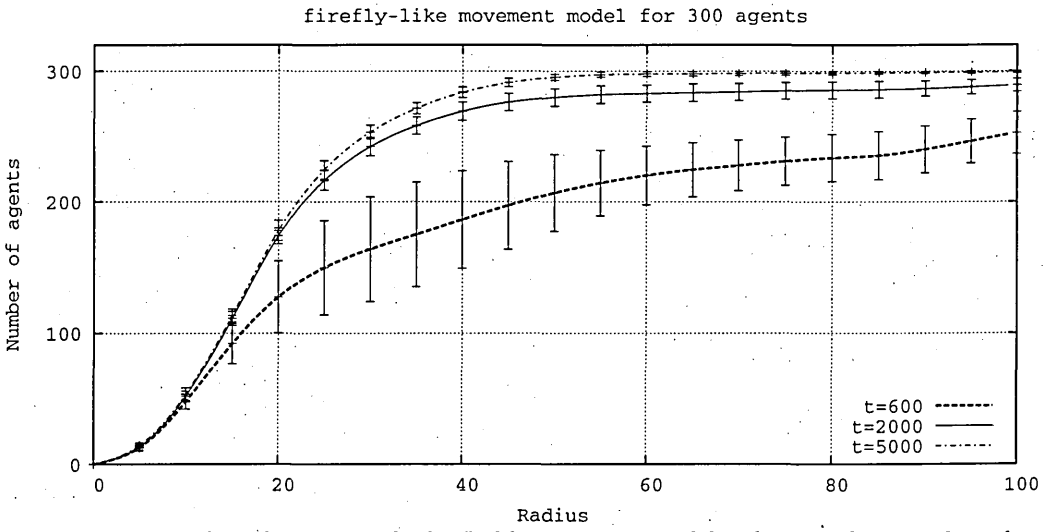


Figure 4.15: Number of agents for the firefly-like movement model within circular area from the centre of APF at different simulation time steps.

4.5.4 Mean distance

In order to further understand the convergence of the swarm, the mean distance, D of each agent towards the *attractor* at each time step during the simulations as in (4.13) has been computed; where x_a and y_a are the x-coordinate and y-coordinate of agent a, and n is the number of agents in the simulation.

$$D = \frac{\sum_{a=1}^{n} \sqrt{(x_a^2 + y_a^2)}}{n} \tag{4.13}$$

The value of the mean distance D, combines two observations from the swarm. First, it will give us an insight on how well spread the agents are around the attractor, and the second is how tight the agents or how close the agents are to each other in the cluster.

Figure 4.16(a)-(c) are the plots of mean distance D, against time for the fish-like, mosquito-like and firefly-like movement models, respectively. Table 4.6 and Table 4.7 shows the mean distance D, for each movement model at t = 400 and t = 5000 simulation time steps. Results show that prior to convergence, the firefly-like movement model exhibit a considerably large variance or standard deviation; as shown in Table 4.6 and the error bars in the Figure 4.16(c). For all the movement models as in the plots of Figure 4.16, when the system reached convergence, the mean distance D increases as the number of agents in the simulation

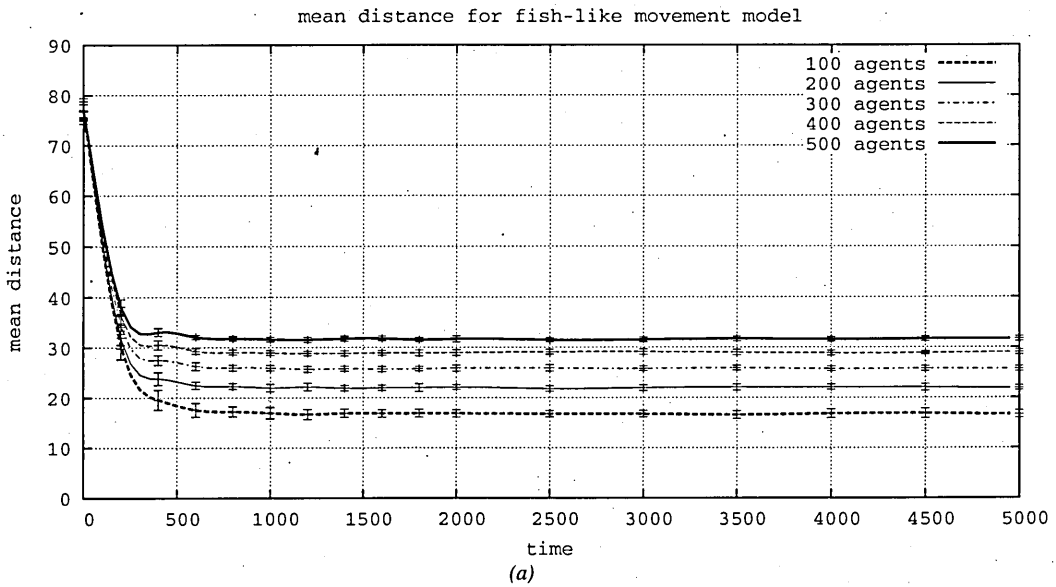
increased. For the firefly-like movement model as shown in the plots of Figure 4.16(c) and Table 4.7 when the system converged, the mean distance D for 300 and 400 agents seems to share the same value of 19.

For ease of comparison, Figure 4.17(a) and (b) show the plots of mean distance D, against time for each movement model with total number of agents of 300 and 500, respectively. As can be observed from the plots, for the fish-like and the mosquito-like movement models, prior to convergence the standard deviations of over 30 runs reaches to about 5; while for firefly-like movement model has a higher standard deviation of around 10 prior to convergence.

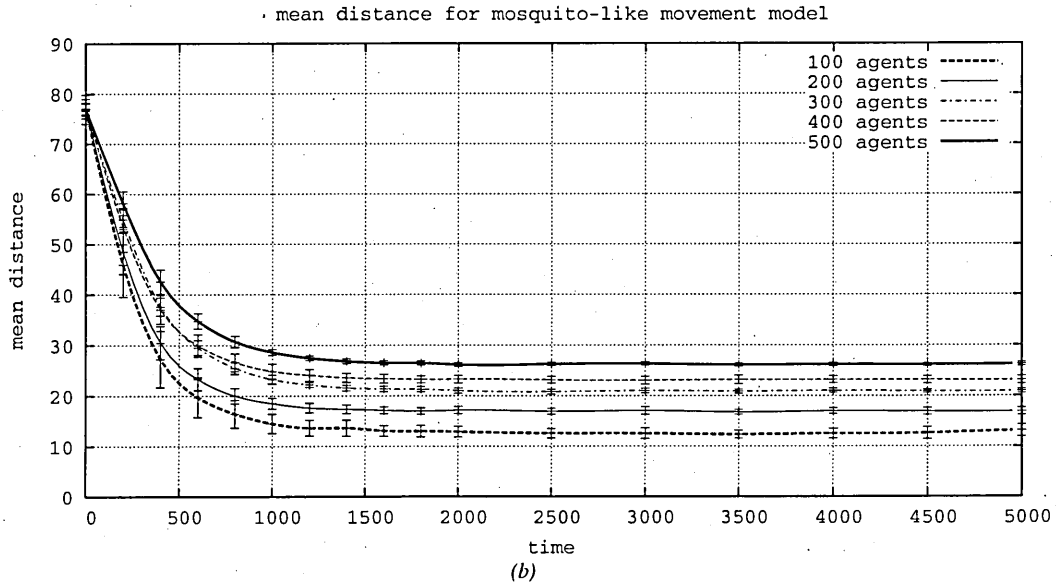
Figure 4.17 also shows the significant difference in convergence rates between the three movement models. The graphs clearly show that the fish-like movement model converges faster than the other two; while the firefly-like movement model is the slowest. This can be explained by the fact that for the fish-like movement model, with a small *movement span* of 10 degrees, agents can cover a wide area in a shorter time; whilst in the firefly-like movement model, with a wider *movement span* of 90 degrees, the agents are more likely to scan within their local area.

From the Figure 4.17(a), it can be seen that the mean distance, D, when the system reached convergence, for the firefly-like movement model is the smallest at around 18 units; while the fish-like model in Figure 4.17(a), has the largest at around 27 units.

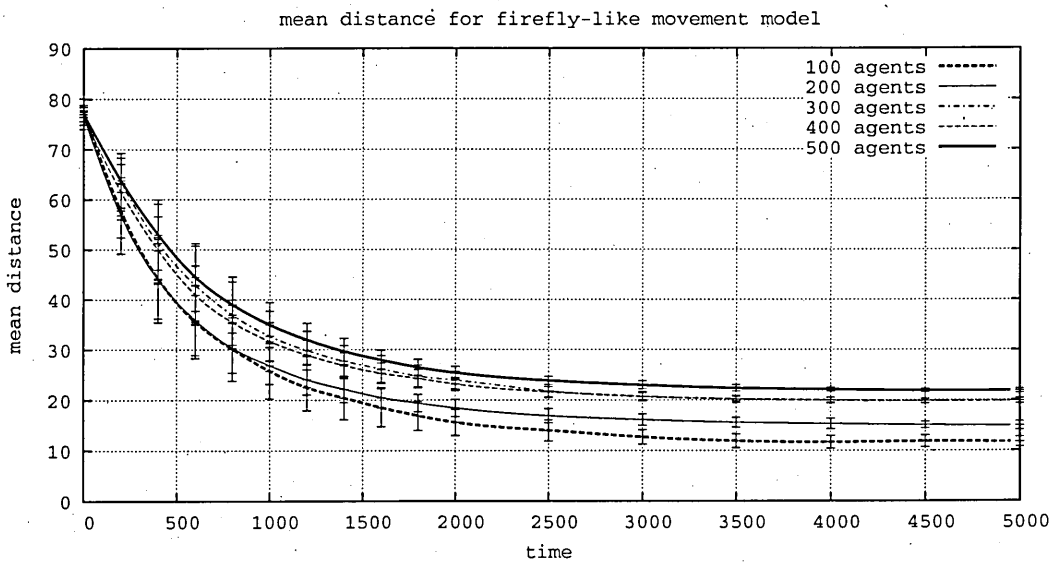
From Figure 4.7(c) for fish-like, Figure 4.10(c) for mosquito-like and Figure 4.13(c) for firefly-like movement models, it can be seen that when the systems converged, they form loose, medium and tight clusters, respectively. It is the innate tendency to form these kinds of clusters that affects the mean distance D values in the plot of Figure 4.17.



(a)

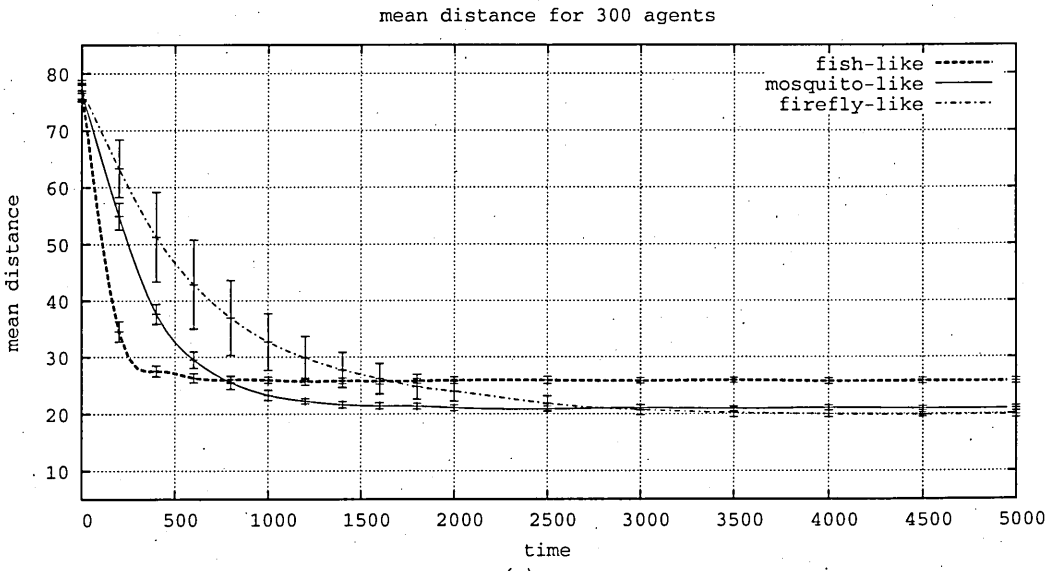


(b)

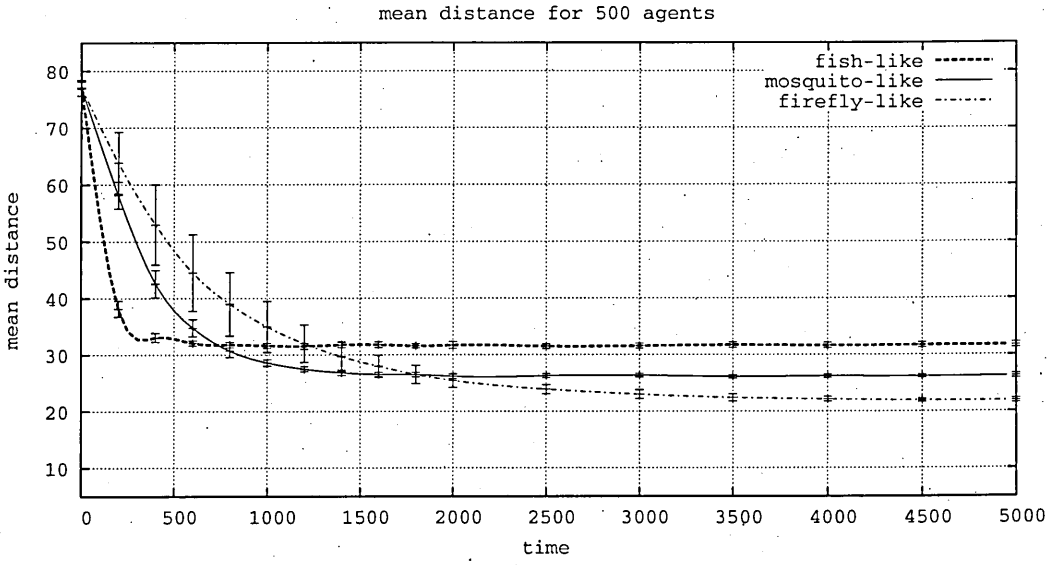


(c)

Figure 4.16: Convergence of mean distance, D for (a) fish-like, (b) mosquito-like, (c) firefly-like



(a)



(b)

Figure 4.17: Convergence of mean distance, D for (a) 300, (b) 500 agents

Table 4.6: Mean distance, D at t = 400 simulation time steps

Total agents	Fish-like		Mosquito-like		Firefly-like	
	mean distance	standard deviation	mean distance	standard deviation	mean distance	standard deviation
100	19.59	1.96	27.26	5.56	44.14	7.94
200	23.81	1.28	30.49	3.31	43.95	8.5
300	27.52	0.95	37.62	1.77	51.26	7.89
400	30.54	0.91	37.06	2.83	49.94	7.88
500	33.07	0.79	42.55	2.39	53.72	6.56

Table 4.7: Mean distance, D at t = 5000 simulation time steps

Total agents	Fish-like		Mosquito-like		Firefly-like	
	mean distance	standard deviation	mean distance	standard deviation	mean distance	standard deviation
100	16.67	0.73	13.09	1.17	11.70	1.00
200	21.90	0.50	16.90	0.71	14.93	0.82
300	25.73	0.52	20.93	0.41	19.83	0.58
400	29.04	0.49	23.13	0.77	19.94	0.74
500	31.70	0.49	26.25	0.40	22.75	1.60

4.6 Summary

The aim of the research in this Chapter was to investigate how a swarm of flocking agents will behave in the presence of an attractive force field in the arena. Many previous studies have concentrated on a repulsive force field. Such works include that of Borenstein and Koren (1989), Kim and Khosla (1992), Khosla and Volpe (1988), and Simmons (1996). These works tend to focus on the same problem in robotics, that of obstacle avoidance. There are also many studies on attractive forces such as that in (Tanner 2003, Hanada 2007, Olfati-Saber & Murray 2003, Esposito & Dunbar 2006, Chen *et al.* 2007, Desai 2002, Mastelone *et al.* 2007, Yang *et al.* 2007, *etc.*). However the attractive forces are only available in the agent-agent cohesion-like strategy. In contrast, the research in this Chapter has examined the agent-object attractive force.

Within this Chapter, Wilensky's (1999) flocking algorithm has been extended and several individual behaviours have been selected in terms of single-agent movement models. An object

which releases an APF is then placed in the centre of the arena and the effect of the APF to the flocking behaviours is studied at a macroscopic level. From the results, it has been shown that by changing the limits of the angle through which an agent can turn, in our case case the *movement span,* various swarming behaviours can be achieved. Several convergence behaviours are also achieved and these behaviours affect the convergence rate in performing an aggregation task.

The flocking model has many applications in the area of robotics and beyond. For example, a group of flocking agents moving together can act as a sensor array, allowing them to locate a desired source in a more effective way. In this Chapter, we have identified, developed and analysed a model for collective movement or flocking in the existence of APF in the arena. Flocking towards an attractor could be useful in information sharing or relay whilst on the move. It is clear that the data from simulations conclude that:

- teams of collective moving agents with a smaller *movement span* are more effective in finding the target (*i.e.* APF) than the larger *movement span.* With collectively moving agents, whenever the APF (*field* > 1.0) is discovered by an agent, the heading of the agent will then be affected by the APF in which it will turn its heading towards the APF slightly. When the agent changes its heading, numerous other agent within its neighbourhood are "pulled in" by local inter-agent influences so that it stays close to the each other.
- collectively moving agents with a larger *movement span* tend to stay close to each other regardless of the APF. Whenever an agent finds a neighbour, it will try to change its heading towards the neighbour (obeying the cohesion strategy). The larger the *movement span* is, the larger the *max-cohere-turn* becomes; resulting in the agents having larger permissible turning angles and allowing the agent to turn towards its neighbour quicker.

Chapter 5 L-Systems for Formation Tasks

One of the main problems in swarm robot systems is that of communication, which requires high bandwidth due to the many-to-many communication between agents. This directly impacts on the ability to form complex patterns. Many previous studies in the field of robot swarms have concentrated on two simple tasks: aggregation and coordinated motion. However, to date, these robots are not able to move and form patterns in a complex way.

The research in this Chapter proposes that by using evolutionary L-Systems, more complex pattern formations in robot swarms can be achieved, provided each agent has the ability to interpret short strings of L-Systems that form the basic DNA of the formation. L-Systems has been studied extensively in the field of computer graphics and so this research presents the first introduction of the use of L-Systems into the area of robot swarm formation. By using L-systems the path between two locations can be represented which can later be used by mobile agents to form an arrangement along the path. In addition, the technique can also be expanded into a path planning algorithm.

5.1 Introduction

The beauty of natural patterns has, for decades, attracted the attention of many researchers. With technological advances, particularly in computer graphics, computer simulations can play an important role for researchers to understand these formations and structures of these patterns. In the field of biological systems, Prusinkiewicz (Prusinkiewicz & Lindenmayer 1990) is believed to be one of the first to model and visualise the growth of tree-like structures.

In nature, many ants species lay trails of pheromones in order to attract other ants while foraging. Laying pheromones is a good strategy in finding the shortest path between the nest and the food source (Bonabeau *et al.* 1999).

When designing large scale multi-agent systems, or swarm systems, an inherent question that needs to be addressed is one of organisation. Agents in the system should be able to form and organise themselves around complex patterns which are generally required to perform specific tasks in a complex arena.

Many previous studies in the field of swarm robotics have concentrated around two tasks: aggregation (Dorigo *et al.* 2004a) and coordinated motion (such as leader-follower)(Othman *et al.* 2005). These robots are not able to move and organise in a complex way. We postulate that this is due to the fact that there is insufficient complexity in the representation of the systems themselves. However, previous representation methods such as graph schemes (Bayazit *et al.* 2002), defeat the challenge of swarm organisation by requiring high communication bandwidth.

One of the requirements of of mobile agents in a swarm is the need to form an arrangement along a path or bridging formations that connect multiple locations. Here, the many path planning algorithms can be used as well, where agents are needed to form an arrangement along the specific path. In this case, the representation of the path is needed to be fed to the agents. One of the methods in representing paths is by using strings in a Logo-style (Abelson & deSessa 1982) format. However this research proposes that the same paths can also also be represented by Lindenmayer Systems (Lindenmayer 1968) with shorter string length, which in the end will save the communication bandwidth between the controller and mobile agents.

Many self-organised path formations algorithms are readily available for multi-agent systems and these have been discussed in Chapter 2, such as Random Growing Tree (RGT) (Avrutin *et al.* 2007), Cyclic Directional Pattern (CDP) (Nouyan *et al.* 2006) and Virtual Pheromones (Payton *et al.* 2004). However, self-organisation is not always the best answer for every problem. In some cases, an alternative method might be preferable. For example, consider that

in an arena a minimum number of agents that are needed to form an arrangement along the path between two locations are present. The self-organised algorithms (RGT, CDP, Virtual Pheromones) might take forever to form an arrangement along the path, and it might not be able to complete the task within the permissible period at all. For that reason it is useful to devise an alternative algorithm even if it is not fully automated or self-organise.

The technique developed in this Chapter proposes that for more complex pattern formations, the level of agent complexity should be increased, albeit marginally. In doing so, one of the basic themes of swarming, *i.e.* limited communication should be retained, is still adhered to. In order to achieve this, the agent should have the capability to transfer information consisting of short bitstrings to its immediate neighbours. It should also have the processing capability to interpret these bitstrings that form the basic DNA of the formation. The transference of short pieces of information is analogous to trophallaxis as a means of communication amongst insects like bees and ants.

The technique developed within this Chapter is to assist multi-agent systems to form an arrangement along the path of two locations, as a communication bridge between two separated points for example. In this instance, agents are needed to make a formation along the path and the information is then transmitted from one end to another using the agents in-between as a medium. Furthermore, as the technique developed within this Chapter uses L-Systems (which uses Logo-style representations), it can also be used as a new approach to path planning algorithms.

In this Chapter we shall fuse ideas developed in the area of computer graphics with that of robotics systems. We introduce a general model for organisation based on Lindenmayer Systems (Lindenmayer 1968), with the addition of an evolutionary algorithm for pattern optimisation. Lindenmayer Systems, or L-Systems for short, provide a symbolic representation of complex dynamic patterns, which were originally used to model biological growth. Evolutionary adaptation of L-Systems alone is not a new idea but we shall show how we can evolve specific formations that can be used to guide the multi-agent system into performing

complex formation-type tasks.

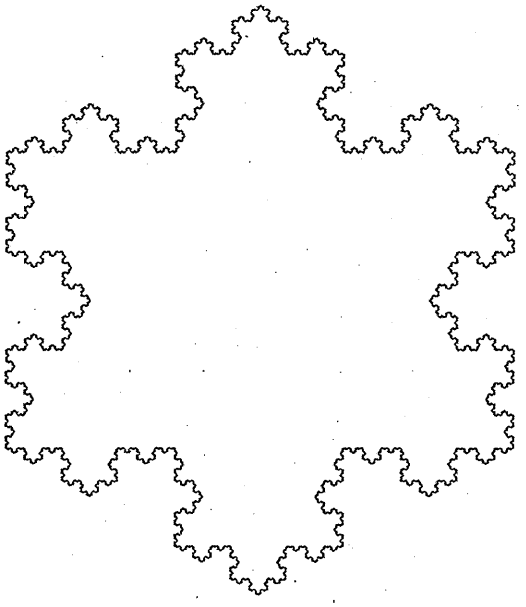
In the following section, the background surrounding the proposed technique is first described, followed by the methodologies and implementations. Next the evaluation of the technique will be described. The results of the simulations are then presented and discussed.

5.2 Background

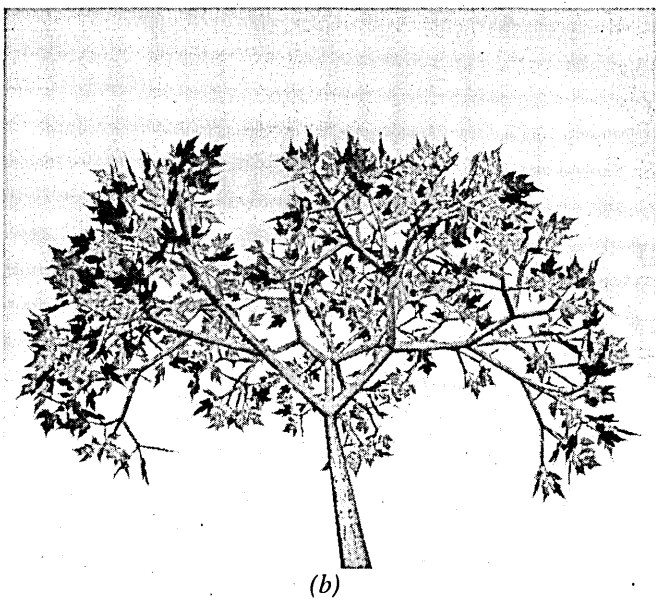
5.2.1 L-Systems

In 1968, the theoretical biologist Aristid Lindenmayer (1968) proposed L-Systems; a mathematical formalism as a foundation for an axiomatic theory of biological development. As a biologist who studied the growth pattern of various types of multi-cellular microorganisms, Lindenmayer at first, devised the L-Systems to provide a formal description of the development of the microorganisms, and also to illustrate the neighbourhood relationship between cells. The system was then extended to describe bigger and higher order plants with complex branching structures. Later in the 1980's, L-Systems found several applications in computer graphics; the two main areas of application are the generation of the fractals (Smith 1984) and the realistic modelling of plants (Prusinkiewicz & Lindenmayer 1990).

L-Systems are considered as one of the "generative grammars" from the "formal grammar" family or sometimes simply referred as a "grammar family". A formal grammar in computer science is a description of a formal language which has a set of strings. Formal grammar can be divided into two main categories; *analytic grammar* and *generative grammar.* An analytic grammar contains sets of rules of how a string can be analysed to determine whether or not it is a member of a particular language, while on the other hand generative grammar contains sets of rules that annotate how strings in a language can be generated.



(a)



(b)

Figure 5.1: Example of patterns generated by L-Systems. (a) Outline of Koch island or snowflake fractal after five iterations of rewriting. (b) Realistic modelling of Fall trees (image copyright of Svetlin (Alex) Bostandjiev of University of California in Santa Barbara)

In the same manner of (Chomsky type) formal grammars, L-Systems generate strings of symbols by repetitively substituting predecessors of given productions by their successors. The basic idea of these grammars is to define complex objects or words by replacing parts of a simple object through a set of rewriting rules or productions. These rewriting process can be carried out recursively. However the main difference between (Chomsky type) formal grammars and L-Systems is that, in Chomsky grammars, productions are applied sequentially,

i.e. one at a time. Meanwhile in the case of L-Systems, productions are applied concurrently to all symbols in a given string. This difference reflects the biological motivation of L-Systems where productions are intended to capture cell divisions in multi-cellular organisms in which many divisions may occur at the same time.

L-Systems can be classified in many different ways, such as:

- Context sensitive (IL-Systems) and context free (0L-Systems).
  - Rules in context free L-System depends only on a single symbol.
  - Rules in context sensitive L-Systems depends on a single symbol and its neighbours.

- Deterministic (DL-Systems) and non-deterministic L-Systems.
  - The L-System is consider as deterministic if there is exactly one production for one symbol, otherwise it is non-deterministic.

- Propagative (PL-Systems) and non-propagative L-Systems.
  - There are at least two symbols needed for the successor of a L-System to be considered as a propagative L-System, if there is only one symbol for the successor, then it will be considered as non-propagative L-System.

- Parametric L-Systems.
  - Parametric L-System operates on parametric words, which are strings of modules consisting of their symbolic names with associated parameters.

These types of L-Systems can be combined. For example a D0L-System (where '0' stands for "0-sided" or "0 context") is a deterministic context free L-System; a PIL-System is a context sensitive with propagation; and so forth. Above all, D0L-Systems are the simplest type of the L-Systems.

The processes in an L-System can simply be divided into two parts: a generative and an interpretative process. The main idea behind the generative process is the string rewriting process.

The generative process

Consider a D0L-System, which can be defined as a triplet G = (**S**, **P**, α), where **S** is a general symbol (a finite non-empty set of symbols); α is the initial start word which usually referred to as the axiom or seed and it is an element of **S**; **P** is a set of production rules of the form of **A** → **x** (predecessors → successors), where **A** ∈ **S** is a symbol in the alphabet and **x**∈**S*** is a (possibly empty) string or word of symbols in the alphabet. Every symbol appears exactly once at the left of a production rule and this makes the system deterministic. As an example, let us consider the following D0L-System:

$G = (\mathbf{S}, \mathbf{P}, \alpha)$

$\mathbf{S} = \{F, R, L\}$

$\alpha : F$

$p_1 : F \rightarrow FRF$

$p_2 : R \rightarrow FL$

$p_3 : L \rightarrow L$

The D0L-System is represented by *F*, *R* and *L* with the axiom represented by the letter *F*. For each letter we specify a rewriting rule or production rule. The rule $F \rightarrow FRF$ means that the predecessor letter *F* is to be replaced by the successor string *FRF*, the rule $R \rightarrow FL$ means that the predecessor letter *R* is to be replaced by the successor string *FL*, and the $L \rightarrow L$ means that the letter *L* will remain as it is. The rewriting process starts from a string called the axiom, in our case it consist of a single letter *F*. In the first generative process, the axiom *F* is replaced by *FRF* using the production $F \rightarrow FRF$. In the second step, the word FRF consist of two letters, both of which are simultaneously replaced in the next generative process. Thus *F* is replaced by *FRF*, *R* is replaced by *FL*, and the string *FRFFLFRF* results. In a similar way, the simultaneous replacement of all the letters will generate the following sequence of words:

$\alpha : F$

$P^1(\alpha) : FRF$

$P^2(\alpha) : FRFFLFRF$

$P^3(\alpha) : FRFFLFRFFRFLFRFFLFRF$

$P^4(\alpha) : FRFFLFRFFRFLFRFFLFRFFRFFLFRFLFRFFLFRFFRFLFRFFLFRF$

...

The interpretive process

In the second part of the L-System, the symbols from one or multiple iterations of string are interpreted and visualised. There are several ways to visualise the L-Systems, one of them is by using the Turtle graphics method.

The Turtle graphic system was created by Seymour Papert in 1960's (Abelson and deSessa 1982). The graphic is the trail left by a moving invisible "turtle", with a state defined by its position and direction. The state of the turtle may change as it moves a step forward, or as it turns at a given angle in the same position. A state of the turtle is defined as a triplet as follows:

(x, y, φ),

where x and y represent Cartesian coordinates of the turtle's position, and the angle φ is the heading or the direction that the turtle is facing. Given the step size d and angle increment Θ, now let us reconsider the previous example of the D0L-System which consists of the three following symbols:

$\mathbf{S} = \{F, R, L\}$.

Given the step size d and angle increment Θ, now the turtle can respond to the following interpretive rules:

- F The turtle moves one step forward in the current direction it is facing leaving a visible trail on the ground by length of d. The state of the turtle changes to (x', y', φ); where $x' = x + d \cos \varphi$ and $y' = y + d \sin \varphi$.
- R The turtle will turn or rotate to the right by angle Θ. The state of the turtle changes to (x, y, φ'); where $\varphi' = \varphi - \Theta$.

- L The turtle turns to the left by angle of Θ. The next state of the turtle changes to (x, y, φ'); where $\varphi' = \varphi + \Theta$.

There are many other rules which can complicate the turtle's graphics and make it possible to generate more complicated pattern. Amongst the most widely used are: :

- Upper case letters other than F, R and L have no graphic representation and the state of the turtle remains unchanged. These letters known as *non-graphical symbols*.
- Lower case of letter *f*, makes the turtle move a step forward by displacement d without drawing a visible trail. By using this rule, it makes possible to construct fractal patterns with unconnected sections. The lower case of letter *f* is usually known as the "moving" symbol.
- An open parenthesis *[* pushes the current state of the turtle onto a LIFO stack; while a close parenthesis *]* pops the top of stack and restores the turtle state. This extension makes branching possible.
- Braces *{ }* indicate that the area that are enclosed in the braces must be filled.

The production rules in D0L-Systems are context free; in other words, the production rules are applicable regardless of the context in which the predecessor appears. In context-sensitive L-Systems, the production rules are dependent on the predecessor's context. For example in 2L-Systems or two-sided L-Systems, the productions will be in the form of $c_L < a > c_R \rightarrow z$, where the strict predecessor letter a can produce string z if and only if the letter a is preceded by letters (or string) c_L and followed by c_R. Thus, letters (or string) c_L and c_R are the left and right *context* of the predecessor letter a. In 1L-Systems, the productions have one-sided context only; the productions can either be in the form of $c_L < a \rightarrow z$ or $a > c_R \rightarrow z$. 0L-Systems, 1L-Systems and 2L-Systems belong to a wider class of IL-Systems, sometimes called *(k,l)*-systems. In a *(k,l)*-system, the left and right context is a word of length k and l letters respectively.

Suppose that we have a new sample of a context-sensitive L-System which has the following axiom, α and productions, **P**:

α : *abbaacc*

$p_1 : b < a \rightarrow b$

$p_2 : a \rightarrow c$

$p_3 : b < b > a \rightarrow c$

$p_4 : b > a \rightarrow a$

$p_5 : a < c \rightarrow a$

$p_6 : c \rightarrow b$

The first few strings generated by the L-System are given below:

α : *abbaacc*

$P^1(\alpha)$: *cbcbcab*

$P^2(\alpha)$: *bbbbbcb*

$P^3(\alpha)$: *bbbbbbb*

$P^4(\alpha)$: *bbbbbbb*

$P^5(\alpha)$: *bbbbbbb*

A context-sensitive L-System (CSL-System) requires that if the neighbours of a symbol match a particular context, then, that symbol should be replaced by the successor symbol. If two rewriting rules apply for a certain symbol, *i.e* one with one-sided context and another with two-sided context, then the one with two-sided context is used. For instance, consider the third symbol from the left in the axiom from the example above. The symbol *b*, is matched with production rules of p_3 and p_4, in this case the production p_3 was used. In general, the rewriting rule that is more specific will overrule the one that is less specific. However, it is possible to encounter conflicts between several rules that can be applied to the same symbol.

5.2.2 Evolutionary algorithms

Genetic algorithms (GA) were first introduced in 1975 by John Holland (Davis 1991). GAs are a class of stochastic search and optimisation techniques based on the evolutionary ideas of natural selection and genetics. The basic ideas of the GA are designed to simulate natural systems processes that are necessary for evolution especially those that follow Charles Darwin's principles of "survival of the fittest". Like in nature, if there is competition amongst individuals

for limited resources, it will result in the fittest individuals dominating over the weaker ones.

The Genetic Algorithm

1. Initialise a population of individuals (chromosomes).
2. Evaluate each individual in the population based on the fitness.
3. Create new individuals by mating current individuals; apply mutation and recombination as the parent individuals mate.
4. Delete members of the population to make room for the new individuals.
5. Evaluate the new individuals and insert them in to the population's pool.
6. If the maximum number of generations is reached, stop and return the best individual; if not, go to step 3.

Figure 5.2: Top-level description of a genetic algorithm

Figure 5.2 shows the anatomy of a general genetic algorithms. The first step of the evolutionary process usually starts from a randomly generated initial population of possible solutions. Members in this population (called chromosomes or genomes) of abstract representations are use to contribute towards the next generation. In each generation (step 2 in Figure 5.2), the fitness for each individual is calculated and evaluated in some way by a fitness function. After the evaluation process has taken place, (step 3 in Figure 5.2) the selection of multiple parent chromosomes for crossover and mutation is performed by randomly selecting from the population, but it is usually influenced by their fitness scores. Some of the old individuals in the population are then replaced with the newly constructed individuals (step 4 and 5 in Figure 5.2). A GA maintains a set of candidate solutions from which it performs a search by iteratively replacing members with poor fitness in the population, with individuals generated by applying variation to fitter members of the population. The GA commonly terminates when either a maximum number of generations has been reached, or a satisfactory fitness level has been produced. If the GA has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

5.2.2.1 Parent selection

The purpose of parent selection in GAs is to increase the reproductive chances of fitter individuals in the hopes that they will produce even fitter offsprings. Many methods for selection exists however in this section we will only give a brief description of Holland's original parent selection method, *i.e.* roulette wheel selection method as described in Figure 5.3. Each individual is assigned with a slice of a circular "roulette wheel", which is proportional to the individual's fitness. The wheel is then spun N times, where N is the total number of individuals in the population. On each spin, the individual which is under the wheel's marker is selected to be in the pool of parents for the next generation. This method can be implemented as in Figure 5.4.

Roulette Wheel Algorithm

1. Calculate the total fitness of all population members; call the result as **total fitness**.
2. Generate n, a random number between 0 and **total fitness.**
3. Select the individual whose fitness, added to the fitnesses of the preceding population individuals is greater than or equal to n.

Figure 5.3: The roulette wheel selection algorithm.

Roulette wheel selection example

Individual	1	2	3	4	5	6	7
Fitness score	8	2	17	7	2	12	11
Running total	8	10	27	34	36	48	59

Random number n	28	2	13	41	31	57	23
Individual chosen	4	1	3	6	4	7	3

Figure 5.4: The roulette wheel selection example. The top table shows the fitness of seven individuals and the running total of fitness. The bottom table shows the individual that would be chosen by the roulette wheel method using these fitness values for each of six randomly generated numbers.

In Figure 5.4, the population consists of seven individuals with a total fitness of 59. The first row in the Figure 5.4 depicts the index of each individual, the second shows the individual's fitness, and the third contains the running total of fitness. Figure 5.4 also shows seven numbers randomly generated between 1 and 59, together with the index of individual that would be selected by roulette wheel parent selection for each of these numbers. In these cases, the selected individual is the first one at which the running total is greater than or equal to the random number *n*. The effect of the roulette wheel selection scheme is to return a set of randomly chosen parents. Although the selection scheme seems to be random, each individual's chance to be selected is proportional to its fitness. After a number of generations, the selection scheme will sideline the least fit individuals and contribute to the spread of the fitter individuals.

5.2.2.2 Crossover

After the selection of parents have taken place, the GA will use the parents to create new individuals or offsprings. Although there are many techniques in creating new offsprings described in the literature, only the traditional methods are described, *i.e.* crossover and mutation operations.

In crossover operations as in Figure 5.5, two individuals are selected as parents. One-point or single-point crossover is the simplest form of crossover operation. The crossover point is chosen randomly. After the crossover point is selected, the parts of two parents after the crossover position are exchanged to form two new offsprings. Crossover operation in GA is extremely important. Many GA practitioners believe that if we remove the crossover operation in GA, the result is no longer a GA (Mitchell 1996).

5.2.2.3 Mutation

Mutation is a GA operator that changes one or more gene value in an individual from its parent, which will result in entirely new gene values being added to the individual. With these new genes, the GA may be able to obtain a better solution. Mutation, as with the crossover operation is an important operation in GA which helps the GA to prevent the population from stagnating

at any local optima. Mutation usually occurs according to a mutation probability, which is always set to a fairly low value. If the mutation probability is set too high, the search will turn into a primitive random search. Figure 5.6 shows an example of the mutation operation. In this example the genes of the parent has been mutated to form a new individual.

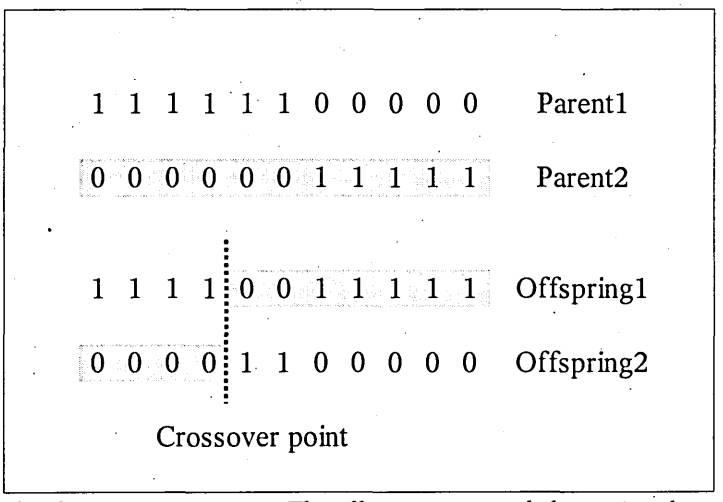


Figure 5.5: Example of one-point crossover. The offsprings are made by cutting the parents at the point denoted by the vertical dotted line and exchanging parental genetic material after the cut.

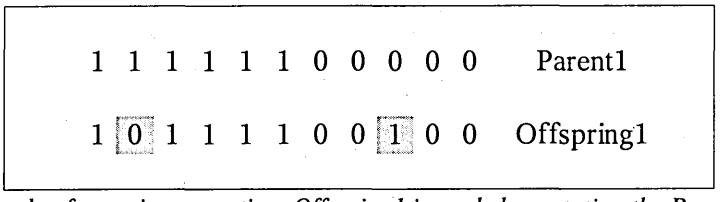


Figure 5.6: Example of mutation operation. Offspring1 is made by mutating the Parent1 at 2nd and 9th bits from left.

5.3 Methodologies and Implementations

As mentioned previously, the proposed system for organised formation of mobile agents consists of:

- an algorithm for pattern formation,
- an algorithm for optimising pattern formation and
- an encoding to represent each agent's position in the arena.

In this thesis, a set of pattern construction commands have been used. These pattern construction commands are used as the input character string for the L-System so that the strings produced by the evolved L-System are a sequence of commands for producing complex patterns.

Once a set of pattern construction commands are generated by the system, it will be passed on to an encoding agent to represent each agent's position (in the arena) for evaluation. After building the pattern, it is evaluated using a multi-part fitness function for how well it fits in the arena and these scores are passed back to the GA engine.

5.3.1 Pattern construction

The pattern construction commands comprises the generative and interpretive module of the L-System as shown in Figure 5.7 and Figure 5.8 respectively. In the generative module, it will take as input an axiom (or seed) α, a set of productions **P** and a set of symbols **S** as inputs. In this work, five symbols namely *F*, *R*, *L*, *[* and *]* as elements of **S** were used to construct the L-System strings, as shown in Figure 5.7 and Table 5.1. *F* represents a forward movement of the turtle in the current direction by 5 units of displacement. *R* and *L* will turn the turtle to the right and to the left respectively, by 25-degrees. Symbols *[* and *]* are the push and pop operators and are used to store and retrieve the state of the current location and direction in the LIFO stack. In the generative module as shown in Figure 5.7, **P** incorporates productions p_1 and p_2. The production p_1 (p_1: $F \rightarrow FRF$) means that in the rewriting process, *F* will be replaced by *FRF*, whilst in the production p_2 (p_2: $R \rightarrow FL$), *R* will be replaced by *FL*. In the rewriting process, the iteration zero represents the axiom α. In the first iteration of the rewriting process, the axiom *F* has been replaced by *FRF* using production p_1. In the second iteration of rewriting process, the product of the first iteration of rewriting process (*FRF*) becomes the subject of the rewriting. The *F* and *R* symbols are replaced with the productions of p_1 and p_2 simultaneously, resulting in the string *FRFFLFRF* after the second iteration of the rewriting process is completed.

Table 5.1: Design symbols and descriptions

Command	Description
[ ]	Push / pop orientation to stack
F	Move forward 5 unit displacement
R	Rotate heading clockwise for 25°
L	Rotate heading counter-clockwise for 25°

Input:

- Axiom, $\alpha = F$
- Elements, **S**: { *F*, *R*, *L*, *[*, *]* }
- Productions, **P**: $\{p_1, p_2\}$
  - p_1: $F \rightarrow FRF$
  - p_2: $R \rightarrow FL$

Rewriting process:

0) *F*

1) *FRF*

2) *FRFFLFRF*

3) *FRFFLFRFFRFLFRFFLFRF*

Figure 5.7: Generative module of pattern construction command.

The interpretive module as shown in Figure 5.8 then constructs patterns by generating a sequence of construction commands that specify how and where the next mobile agent's position would be in the arena relative to itself. This sequence of commands is based on the instruction language for a Logo-style turtle (Abelson and deSessa 1982). A stack is also maintained through the use of 'push' and 'pop' operators. A visualisation of the L-System is also shown in the figure. The module takes a string and a set of interpretation rules as inputs. In this example (Figure 5.8), the string is taken from the previous rewriting process after the 3rd iteration of the rewriting process, and the interpretation rules consist of *F*, *R* and *L*. Here, *F* means move forward; *R* means turn to the right for 90°; and *L* means turn to the left for 90°. For demonstration purposes, in the figure, only the first 15 letters from the string have been

visualised.

Input:

- string: *FRFFLFRFFRFLFRFFLFRF*
- interpretation rules:
  - ➔ *F*: go forward
  - ➔ *R*: turn right 90°
  - ➔ *L*: turn Left 90°

Visual interpretation:

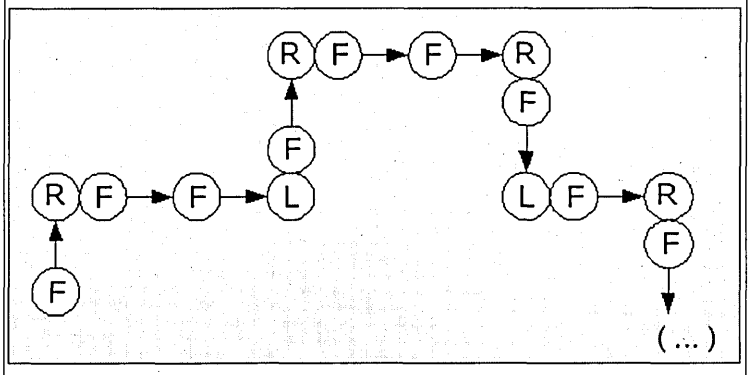


Figure 5.8: Interpretive module of pattern construction command.

In Figure 5.9, the intermediary steps of building a pattern are shown; where red dots indicate the location of agents, and blue lines indicate the parent-child connection of the agents. Figure 5.9(a) is the axiom pattern which has been built from the string *FFRFRFLFLLFF*, while Figure 5.9(b) is the rule string pattern from the string *RFLF*. Figure 5.9(c) is the pattern formed after the first iteration of rewriting the aforementioned axiom and rule using production rule of $F \rightarrow RFLF$.

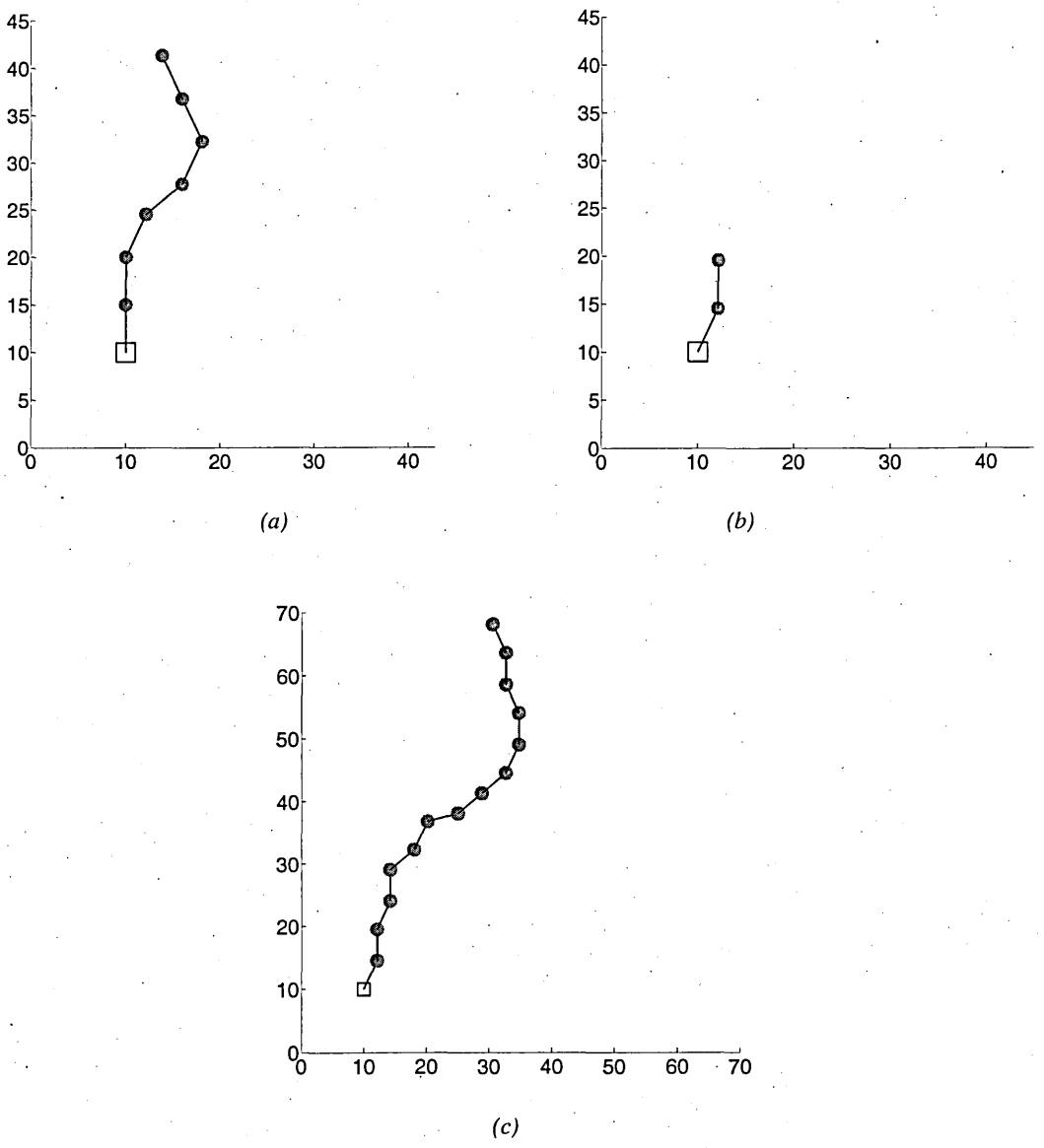


Figure 5.9: Visualisation of L-System: (a) the axiom; (b) the rule string, (c) formed pattern after the first iteration of rewriting process

5.3.2 Representation methodologies

Within this research, two systems; D0L-Systems and CSL-Systems have been designed to represent the patterns formed by the robot swarms. For both systems, the same predecessor is used, *i.e.* the symbol *F*. For D0L-Systems there is one production rule p_1, whilst for CSL-Systems four production rules are used (p_1, p_2, p_3 and p_4). The arrangement of the production rules used in this work is given below:

p_1 : predecessor → $succ_1$

p_2 : $c_L <$ predecessor → $succ_2$

p_3 : predecessor > c_R → $succ_3$

p_4 : c_L < predecessor > c_R → $succ_4$

where p_1, p_2, p_3 and p_4 are the production rules **P**, $succ_1$, $succ_2$, $succ_3$, and $succ_4$ are the successors, while c_L and c_R are the context for the left and the right side of the predecessor respectively. As mentioned in the previous section, during the rewriting process in the generative module, the predecessor will be replaced with a successor accordingly. Let us consider p_2, the predecessor in this case will be replaced with $succ_2$ if the left context (c_L) is met. In p_3, the predecessor will be replaced with $succ_3$ if the right context (c_R) of the predecessor is met. Finally in p4, the predecessor will be replaced with $succ_4$ if and only if the right context (c_R) and the left context (c_L) of the predecessor are met. However, based on the priority, the more specific rewriting rule or production rule will overrule. For instance, if the conditions for p_4 are met, the generative module will use p_4 and will ignore p_1, p_2 and p_3.

5.3.3 Evolutionary algorithms

5.3.3.1 Encoding

An individual L-System is optimised by using an evolutionary algorithm. The initial population of L-Systems is created by randomly creating axioms, successors and contexts. The successor will then replace all the symbols *F* in an axiom string during the rewriting process. The evolutionary process then proceeds by selecting a collection of highly fit individuals as parents and then using them to create a new population by mutation and crossover operations.

An initial L-System (axiom, successors and contexts) is created randomly using a blank template with an arbitrary number of symbols (consisting of *F*, *L*, *R*, *[* and *]*) to be included in the string. For D0L-Systems, the axiom string is between 8 and 12 symbols in length, and the successor string is between 10 and 12 symbols in length. For CSL-Systems, the length of the axiom and successors strings are between 8 and 12 symbols, and between 1 and 3 symbols for the both contexts (c_L and c_R). The reason for limiting it to 12 symbols in length for axioms and successors is that, if the length of an axiom is too long it might defeat the purpose of the rewriting process of L-Systems (where the rewriting process is the core business), if after the

first iteration of the rewriting process, the formed pattern has achieved the goal. Another reason is the intention to "grow" the formation. If the string becomes too long then there is insufficient room to grow the formation. For the contexts (c_L and c_R) where we limit the length for up to 3 symbols is simply that, the longer the context is the harder the condition (for the context) will be met. During any time step of the simulation, the number of symbols F in an axiom string and rule string have been predefined to have a minimum of two and one respectively. This way all the L-Systems will generate patterns after they have been interpreted or visualised.

In using evolutionary algorithms, first the L-Systems are encoded to be a chromosome-like structure. Chromosomes for D0L-Systems are made up of two genes or two parts. The first gene is the axiom (α) and the second is the successor ($succ_1$). While for CSL-Systems, each chromosome are made up from seven elements or genes; an axiom (α), two contexts (c_L and c_R) and four successors ($succ_1$, $succ_2$, $succ_3$ and $succ_4$). The chromosome-like structure for D0L-System, G and CSL-System, H are as shown below:

- G : [α $succ_1$],
- H : [α c_L c_R $succ_1$ $succ_2$ $succ_3$ $succ_4$].

After the L-Systems are created, the rewriting process will be executed for up to 10 iterations or until the number of symbol F reaches 100 or more, whichever comes first. From these solutions, the fitness will be calculated and evaluated once, and the number of rewriting iterations of the fittest individual will be recorded along with other fitness scores. If the fitness of an L-System scores above a preset threshold, the L-System will be passed onto the GA pool which consists of 50 individuals; otherwise it will be discarded, and a new L-System will be created to replace it. By doing this, the initial population of L-Systems will have a variety of solutions with individual fitness values above the preset threshold, thus maintaining a healthy population.

5.3.3.2 Selection method

In this research, as there are several fitness functions which lead to several fitness scores, Goldberg's (1989) Pareto ranking method to rank the L-Systems population with added

modification has been used. Goldberg (1989) suggested *non-dominated sorting* to rank the population according to Pareto optimality. In this scheme, the currently non-dominated chromosomes or individuals in the population are given rank one and removed from the population. Then the population is ranked again, the newly non-dominated individuals in the reduced population are assigned rank two and removed from the population. The process continues until all the members of the original population are ranked.

In this case, different priorities have been set for each goal. For instance, in attempting to meet the goals, $\mathbf{g} = \{g_1, g_2, g_3\}$, where the priority of the former is higher than the latter, *e.g.* priority(g_1) > priority(g_2) and priority(g_2) > priority(g_3). In the first step of ranking process, the procedure suggested by Goldberg is followed by using the non-dominated sorting method for all goals **g**.

In the second step of the sorting process, firstly all the individuals that have been given rank one in the previous step are gathered, and then the group using the same process of non-dominated sorting for goals g_1 and g_2 are ranked, until all the selected individuals have been ranked. Then the second ranked individuals from the previous step are gathered and these are sorted and ranked again for g_1 and g_2 and so forth. In the third step, after all the individuals have been ranked into g_1 and g_2, the population will then be sorted and ranked again into g_1 following the second step procedure.

The next procedure is to decide which individuals in the population will be used to create offsprings for the next generation, and how many offsprings will be created. The purpose of selection is to emphasise the fitter individuals in the population in the hope that their offspring will in turn have an even higher fitness. Mitchell (1996) stated that, selection has to be balanced with variation from mutation and crossover: a selection process that is too strong will result in a suboptimal but highly fit individual that will take over the population; whilst too weak and the selection will result in slow evolution.

Tournament Selection with *Elitism* methods was used in this work. In the Tournament Selection method, two individuals are first chosen randomly from the population. Then a random number m, between 0.0 and 1.0 is generated. If the number m is smaller than n ($m < n$) (where n is a parameter, in our case $n = 0.8$), than the fitter individual will be selected as a parent, otherwise the less fit one is selected. The two individuals will then be returned to the population and can be re-selected again (Goldberg & Deb 1991).

Elitism (De Jong 1975) is a method to force the GA to retain some number of best individuals at each current generation and pass them on to the next generation. The best individuals could be lost if they are not selected to reproduce or if they are destroyed by mutation and / or crossover operations. In this work, we retained 5 individuals as elitists and passed them to the next generation without going through GA operations, such as mutation and crossover.

5.3.3.3 Genetic operators

Mutation (Figure 5.10) and *blending* (Figure 5.11) are used to create new individuals. In blending, parents will be selected randomly. Blending then takes place either in axiom strings or rule strings between two parents and will produce two new individuals.

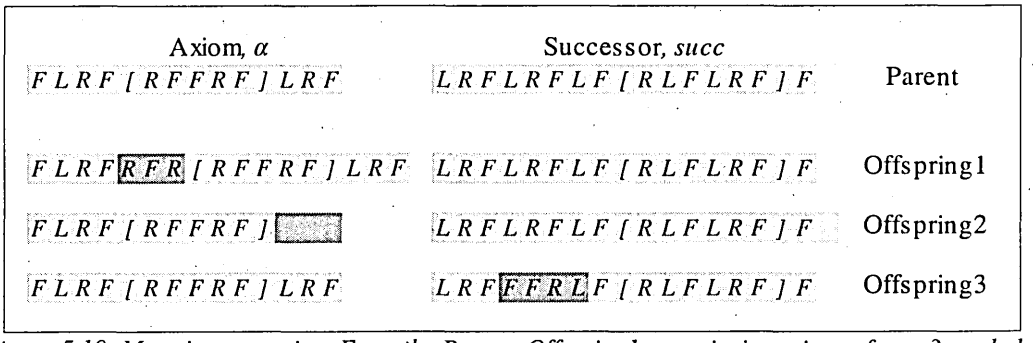


Figure 5.10: Mutation operation. From the Parent, Offspring1 commits insertions of new 3 symbols; Offspring2 deletes 3 symbols in the axiom string; Offspring3 replaces 4 symbols in the successor string.

In mutation, after selecting the parent, the axiom or any successor will be mutated in a predefined way. Changes in mutation that can occur include:

- inserting one or more symbols in random locations of the string, or
- deleting one or more symbols in the selected string, or

- replacing one or more symbols with random symbols.

For instance, let us consider the D0L-System as in Figure 5.10; if the L-System L1 (axiom string $_{L1}\alpha$, successor string $_{L1}succ_1$) is selected to be mutated,

$_{L1}\alpha$: *FLRF [RFFRF] LRF*

$_{L1}succ_1$: *LRF LRFL F [RLFLRF] F*

Some of possible mutations are,

- insert new symbol in the string, offspring1

  $_{L1}\alpha$: *FLRF **RFR** [RFRFRFRF] LRF*

  $_{L1}succ_1$: *LRF LRFL F [RLFLRF] F*

- delete random symbols, offspring2

  $_{L1}\alpha$: *FLRF [RFFRF]* ____

  $_{L1}succ_1$: *LRF LRFL F [RLFLRF] F*

- replace symbols, offspring3

  $_{L1}\alpha$: *FLRF [RFFRF] LRF*

  $_{L1}succ$: *LRF **FFRL** F [RLFLRF] F*

For an example in blending, as shown in Figure 5.11 let us consider D0L-Systems; two parents, L2 ($_{L2}\alpha$, $_{L2}succ_1$) and L3 ($_{L3}\alpha$, $_{L3}succ_1$) were selected randomly and be used to create two offsprings, LC2 ($_{LC2}\alpha$, $_{LC2}succ_1$) and LC3 ($_{LC3}\alpha$, $_{LC3}succ_1$). First, LC2 will make a copy of L2, and it will then insert a small part of L3 into it. This is done by replacing several symbols either from axiom string or successor string from L3, to either axiom string or successor string of L2 to become LC2.

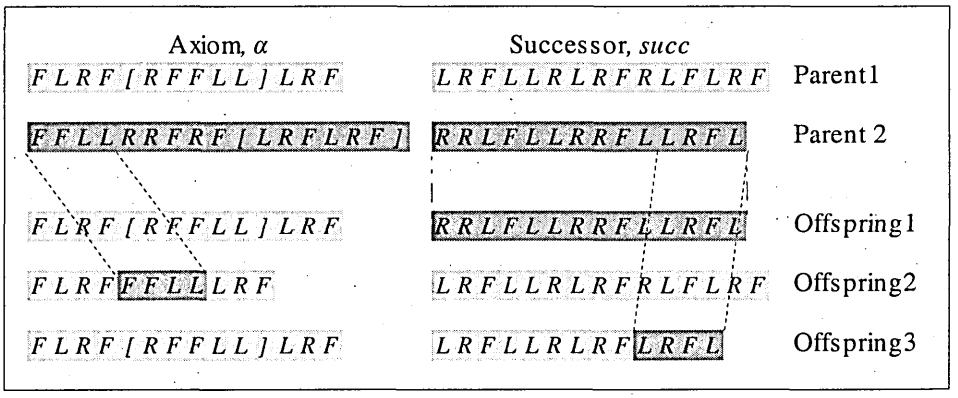


Figure 5.11: Some of possible blending operations. At first, all the offsprings makes a copy of Parent1. Offspring1 replaces the successor string taken from Parent2; Offspring2 replaces some of its symbols in the axiom string and replaces with some symbols taken from Parent2's axiom string; Offspring3 takes and replaces some symbols in its successor string with some symbols taken from Parent2's successor string.

For example if the parent L2 with the following axiom $_{L2}\alpha$ and successor $_{L2}succ_1$ is given by:

$_{L2}\alpha$: *FLRF [RFFLL] LRF*

$_{L2}succ_1$: *LRFL LRLRF RLFLRF*

and the parent L3 with the following axiom $_{L3}\alpha$ and successor $_{L3}succ_1$, are selected, *i.e.*

$_{L3}\alpha$: *FFLL RRFRF [LRFLRF]*

$_{L3}succ_1$: *RRLF LLRRF LLRFLF*

then the possible blending results of LC2 ($_{LC2}\alpha$, $_{LC2}succ_1$) are as follow:

- replace entire in successor string $_{LC2}succ_1$ with $_{L3}succ_1$:

  $_{LC2}\alpha$: *FLRF [RFFLL] LRF*

  $_{LC2}succ_1$: ***RRLF LLRRF LLRFLF***;

- replace several symbols in axiom $_{LC2}\alpha$ taken from $_{L3}\alpha$:

  $_{LC2}\alpha$: *FLRF* ***FFLL*** *LRF*

  $_{LC2}succ_1$: *LRFL LRLRF RLFLRF*

- replace several symbols in successor string $_{LC2}succ_1$ taken from $_{L3}succ_1$:

  $_{LC2}\alpha$: *FLRF [RFFLL] LRF*

  $_{LC2}succ_1$: *LRFL LRLRF* ***LRFL***.

Another operation of evolutionary algorithms used in this work is the crossover operation. We perform crossover operations in two different ways as follow:

- one is by swapping the gene(s) or element(s) in the L-Systems chromosomes as shown in Figure 5.12,
- another is by swapping symbol(s) from the specific gene(s) of two different parents as shown in Figure 5.13.

For example in Figure 5.12, let us consider two parents from CSL-Systems, namely L4 and L5 with the following chromosomes:

$$L4 : [\ {}_{L4}\alpha\ {}_{L4}C_L\ {}_{L4}C_R\ {}_{L4}succ_1\ {}_{L4}succ_2\ {}_{L4}succ_3\ {}_{L4}succ_4]$$

$$L5 : [\ {}_{L5}\alpha\ {}_{L5}C_L\ {}_{L5}C_R\ {}_{L5}succ_1\ {}_{L5}succ_2\ {}_{L5}succ_3\ {}_{L5}succ_4]$$

are selected to create two children of LC4 and LC5. Then the possible crossover results are as follow:

- one gene swapping,

$$LC4 : [\ {}_{L4}\alpha_1\ {}_{L4}C_L\ {}_{L4}C_R\ \underline{{}_{L5}succ_1}\ {}_{L4}succ_2\ {}_{L4}succ_3\ {}_{L4}succ_4]$$

$$LC5 : [\ {}_{L5}\alpha_1\ {}_{L5}C_L\ {}_{L5}C_R\ \underline{{}_{L4}succ_1}\ {}_{L5}succ_2\ {}_{L5}succ_3\ {}_{L5}succ_4]$$

- two genes swapping,

$$LC4 : [\ {}_{L4}\alpha_1\ \underline{{}_{L5}C_L}\ {}_{L4}C_R\ {}_{L4}succ_1\ {}_{L4}succ_2\ \underline{{}_{L4}succ_3}\ {}_{L4}succ_4]$$

$$LC5 : [\ {}_{L5}\alpha_1\ \underline{{}_{L4}C_L}\ {}_{L5}C_R\ {}_{L5}succ_1\ {}_{L5}succ_2\ \underline{{}_{L5}succ_3}\ {}_{L5}succ_4]$$

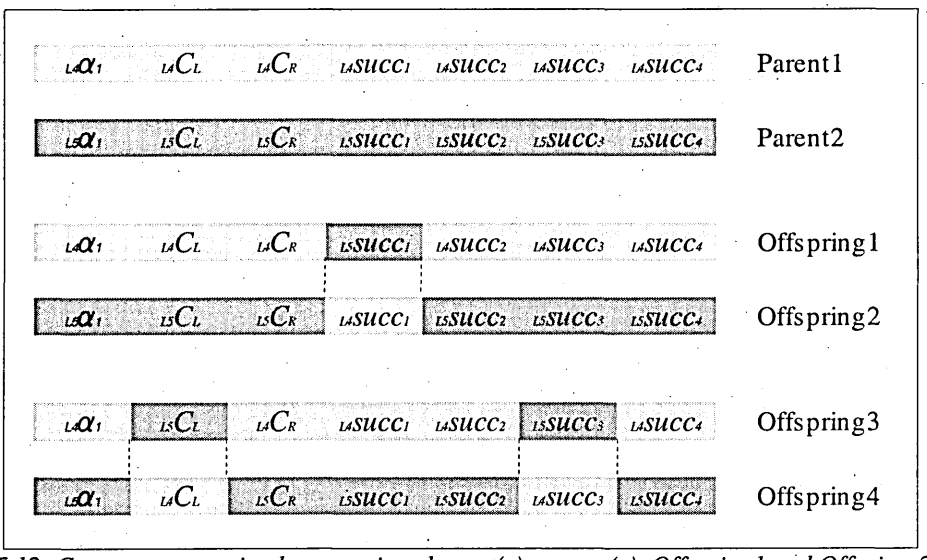


Figure 5.12: Crossover operation by swapping element(s) or gene(s). Offspring1 and Offspirng2 depict one gene swapping, and Offspring3 and Offsping4 show 2 genes swapping take place.

The second way of the crossover operation is by swapping symbols from two genes of two different parents. For instance in Figure 5.13, if we have two parents of CSL-Systems, namely L6 and L7, and chosen $succ_2$ genes from both parents as follow:

$_{L6}succ_2$: *LRFL RFLF [RLFLR] FRRF*

$_{L7}succ_2$: *FLRF [RFFLL] LRFR LFLRF*

to perform a crossover operation. Some of possible solutions are as follow:

- one-point crossover,

  $_{LC6}succ_2$: *LRFL RFLF* ***LRFR LFLRF***

  $_{LC7}succ_2$: *FLRF [RFFLL]* ***[RLFLR] FRRF***

- two-point crossover,

  $_{LC6}succ_2$: *LRFL* ***[RFFLL] LRFR*** *FRRF*

  $_{LC7}succ_2$: *FLRF* ***RFLF [RLFLR]*** *LFLRF.*

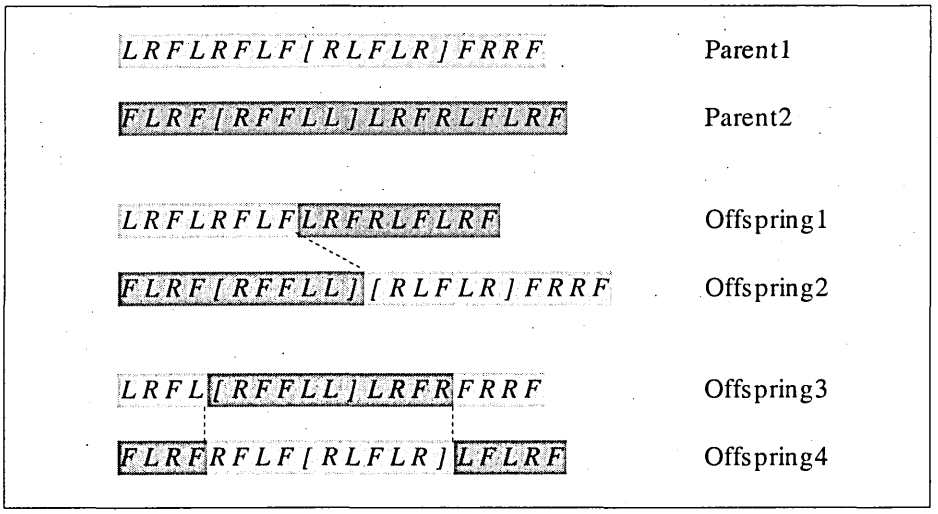


Figure 5.13: Crossover operation by swapping symbols. Offspirng1 and Offspirng2 show example of one-point crossover; Offspring3 and Offsprig4 show the example of two-point crossover.

5.3.4 Evolving the patterns - pre-runs

With all the ingredients described above, pattern formation of robot swarms can be modelled by using the L-System evolution process. In order to evolve patterns, the first thing that needs to be done is to define a task and the fitness functions. The simplest pattern formation for multi-agent systems is the exploration task. In this case, agents are expected to fan out along some criteria. Figure 5.14 shows an example of an L-System that has been evolved for exploration in

an open arena, *i.e.* an arena without any obstacles. The arena size in this example has been set to 100 by 100 unit square. Red dots indicate the location of agents, and blue lines indicate the parent-child connection of the agents. In this example, the population size of the GA is 50. Two production schemes are used. In the first reproduction module, elite parents are selected for crossover at rate of 33%. In the second reproduction module, random parents are selected with a crossover rate of 60%.

At first glance it may seem that the pattern is somewhat random. However in reality, the evolved string, *i.e.* the chromosome that represents the pattern, is regular and assumes the shape shown after iterating through three times, with a preset axiom.

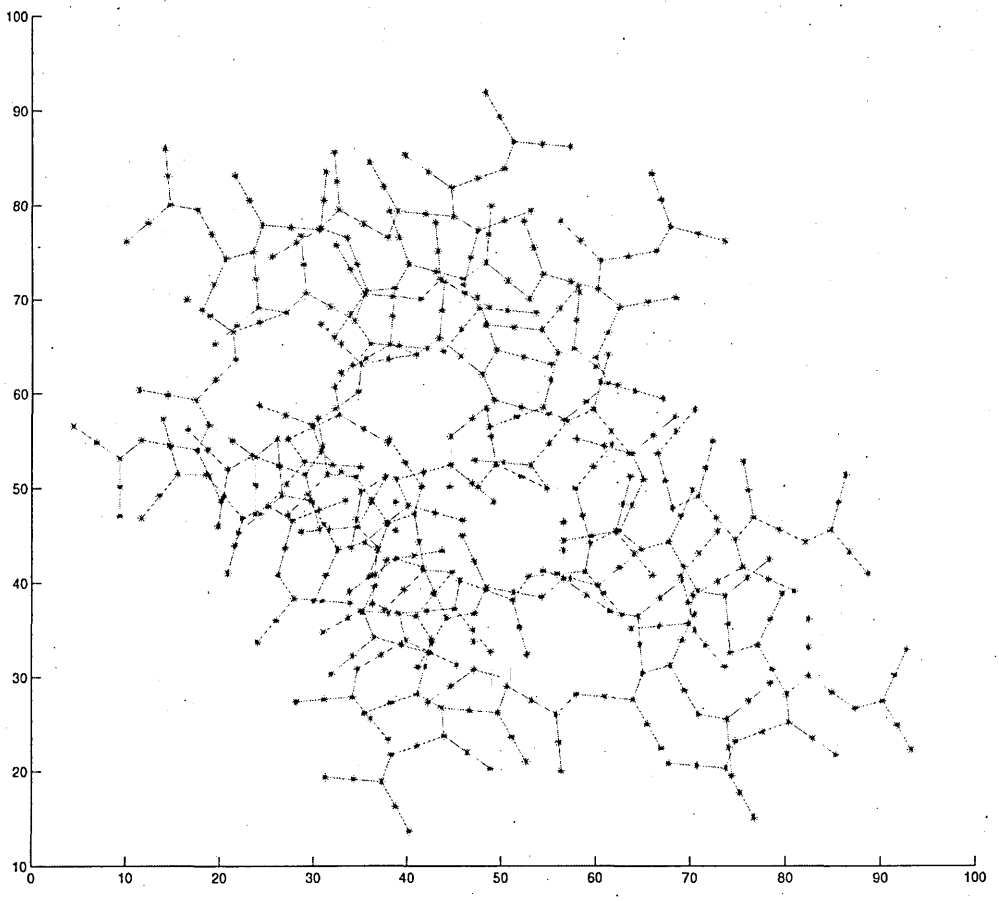


Figure 5.14: Example evolution of an L-System that maximises spread for exploration purposes. Each red point indicates the location of an agent, and the blue lines indicate parent-child relationship.

5.4 Evaluation / Simulation

A series of simulations have been carried out to evaluate our pattern formation algorithm approach under different experimental conditions, *i.e.* the arrangement of the obstacles, and to compare them. In particular, we considered the working arena to be in multiple levels of difficulty, obtained by the various standard obstacle(s) arrangements in the arena. In the following sub-sections, the evaluation procedure is specified. The simulation methodology is briefly described and finally the results will be presented and discussed.

5.4.1 Task and procedure

In this research, the topic of interest is how the pattern formations of robotic swarm can be represented by (evolved) L-Systems. With all the basic ingredients in hand, the tasks and the working arena need to be defined. Simulations have been done using a proprietary software from MathWorks Inc. called MATLAB. Three different working arenas, as shown in Figure 5.15, have been defined; namely *open*, *cross* and *scatter*, with a bounded arena of size 200 by 200 units. As their names imply,

- the "open" arena refers to an arena without obstacles. This is often used to test the minimum requirement to connect two points, and to analyse the complexity in path planning research.
- "cross" refers to an arena where there is an obstacle with a cross shape present in the centre arena – the cross serves as a major obstacle between the two points that are to be connected.
- "scatter" refers to an arena where there are obstacles randomly scattered around the arena – this essentially serves to test algorithms in a maze like environment.

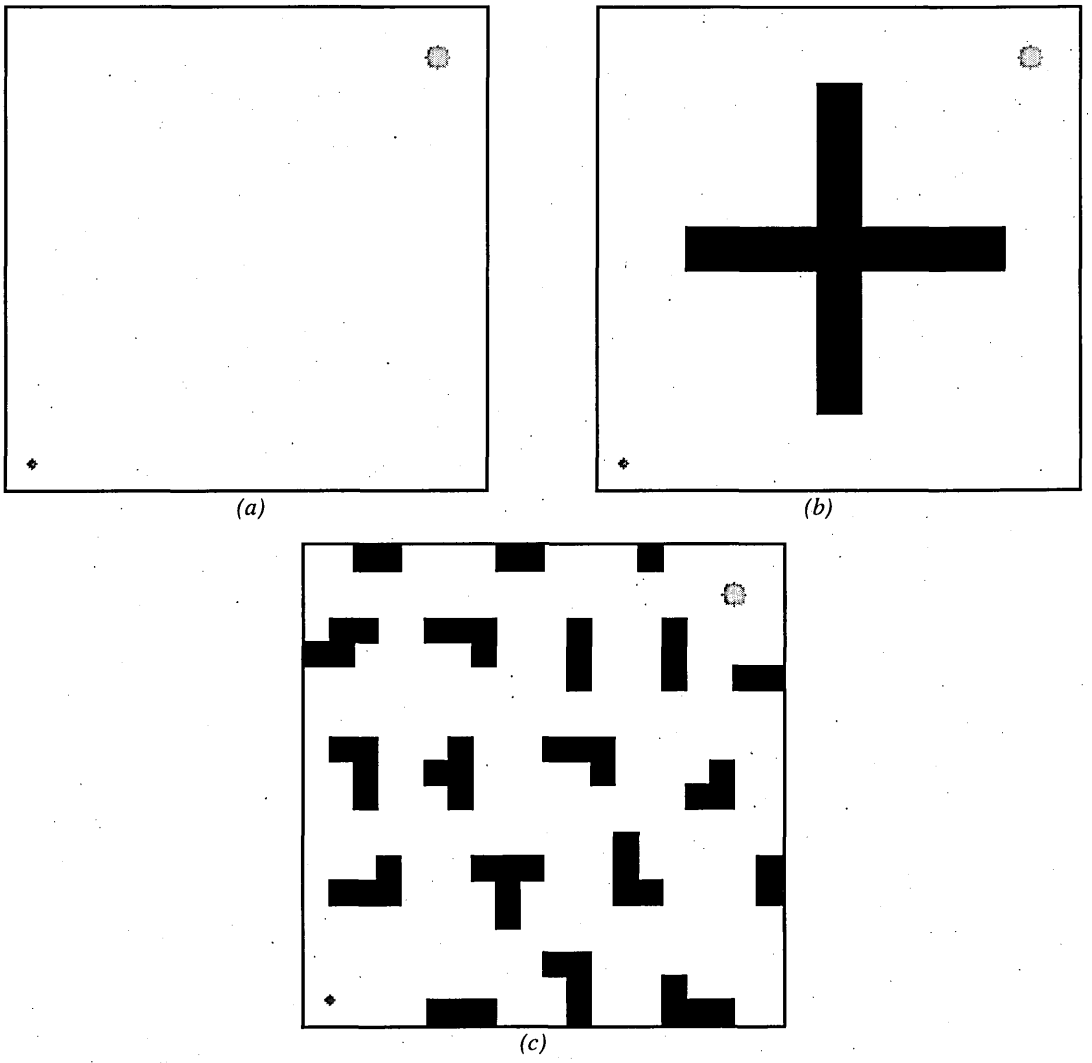


Figure 5.15: Working arenas where the black colour box(es) indicate(s) the obstacle(s): (a) open, (b) cross, (c) scatter. The blue square on the bottom left depicts the start location and the red square on the top left of the arena shows the goal location.

5.4.1.1 Evolutionary process

To evolve patterns, several fitness functions have been defined. The objective is to evolve a formation that connects two locations, *i.e. start* and *goal* as shown in Figure 5.15, while avoiding obstacles (if they exist) in the arena. The start point is at the bottom left corner with the coordinate of (10,10), and the goal location is defined as a square box of 10 unit sides with the centre coordinate of (180, 180).

The fitness of each individual is based on three elements, as follows,

$$fitness = \{ nOut, coverage, d2goal \} \quad (6.1)$$

The first element is the number of agents that reside in the restricted areas such as inside the

area of an obstacle or outside of the the arena (*nOut*). For this element, we seek the number of agents to be minimal or zero. The second element is the agent's coverage in the arena (*coverage*). The agent's coverage is defined by a measure of rectangular area required to enclose or bound all the agents, and we seek to maximise its value. The final element is based on the nearest distance of any agent in the formation to the goal location (*d2goal*). For this element, the closer the individual gets to the goal, the fitter the individual is.

For an L-System to be selected as one of the individuals for the initial population, the L-System has to have a certain degree of healthiness or score above a preset threshold. In this work, the initial threshold is set to five, *i.e.* a maximum of five agents are allowed in the prohibited area such as inside the obstacle or outside the arena perimeter. If the individual scores six in this instance, the individual will be discarded and the new individual will be recreated. This value of five will be optimised downwards to zero as the simulation executes.

5.4.1.2 Piece-wise solutions

As mentioned previously, the evolutionary algorithm has been configured to run with an initial population of 50 individuals with a preset fitness threshold and with a maximum of 100 generations. The evolutionary process will end if either:

- it reaches the maximum number of generation, *i.e.* 100, or
- the fittest individual's fitness scores have been stagnant for some generations.

In this Chapter, the abovementioned stagnancy number has been set to 20. This means that if the fitness scores of the fittest individual stalls for the last 20 generations, the evolutionary process will cease.

If any of the above conditions have been met, but the pattern formed has not reached the goal, the evolutionary engine will then select one of the agents (points) according to some criteria as a next start point. This essentially is used to create a non-continuous solution made up of multiple segments which offers more flexibility. The criteria for an agent (or point) to be selected as a next start position are:

- the agent should be the nearest agent to the goal, and

- the agent is not too close to the obstacles.

In this work, the parameter that measures closeness to an obstacle is defined as 5 units displacement.

After a next start point has been selected, the evolutionary engine will start evolving the next piece or segment of the L-System until one of the agents reaches the goal, or until it has reached the maximum number of generations.

The aforementioned method has been tested with D0L-Systems and CSL-Systems. The method can be summarised as follow:

1. randomly generate 50 L-Systems,
2. evaluate and evolve the L-Systems for up to 100 generations,
3. if the formation has not reach the goal, then do the piece-wise solutions.

In the next subsection, the results from the simulations are presented, and the output is discussed.

5.4.2 Results

The following results are collated by simulating the task of generating a formation connecting two locations, namely *start* and *goal* as previously described. All simulations uses a square arena of size 200 by 200 units with three different obstacles arrangement in the arena. Fifty one runs are made for each model (D0L-Systems and CSL-Systems) and each arena. The fittest individuals' data for analysis were recorded at every generation during the simulation.

Figure 5.16, Figure 5.17 and Table 5.2 provides an overview of the overall performance of the proposed model. Figure 5.16 and Figure 5.17 show the graphs from 51 runs in the each one of the arena arrangements for D0L-Systems and CSL-Systems respectively. From the graphs and data (Table 5.2) obtained, the overall performance of D0L-Systems is better, in the sense that the number of non-continuos segments of the L-Systems are lower compared to the one exhibited by the CSL-Systems. For D0L-Systems, more than 50% of the simulation runs, regardless of the arena arrangement, evolves into 2 or less segments for the formed pattern. For

the CSL-Systems however, even in the open arena, around 50% of the simulation runs ended up with having one segment for the L-Systems.

Table 5.3 and Table 5.4 are the tabulated data for the average of the total number of agents that formed the pattern and its standard deviations for D0L-Systems and CSL-Systems respectively with regard to the number of segments of the L-Systems. The results clearly show that as the number of segments increases, the number of agents increases accordingly. The difference between the D0L-Systems and the CSL-Systems is small, when considering the total number of agents in the formations.

Figure 5.18, Figure 5.19 and Figure 5.20 show the the plot of the number of agents, coverage and the nearest distance to the goal respectively against generations during the evolutionary process from one of the D0L-Systems samples in the scatter arena. The plots are based on the fittest individual of every generation. In this instance, the D0L-System successfully formed the pattern by connecting two locations with only one segment, at the point that the simulation ended at 100^{th} generation. From the plot in Figure 5.19, it clearly shows that the coverage (rectangular area required to enclosed or bound all the agents) increases as the number of generations increases where the goal is to maximise the area coverage in order to "grow" the formation. Figure 5.20 is the plot of the nearest distance to the goal. As the number of generation increases, the nearest distance of one of the agent to the goal decreases. From the plot we can see that the formation reaches the goal at around the 85^{th} generation of the evolutionary process.

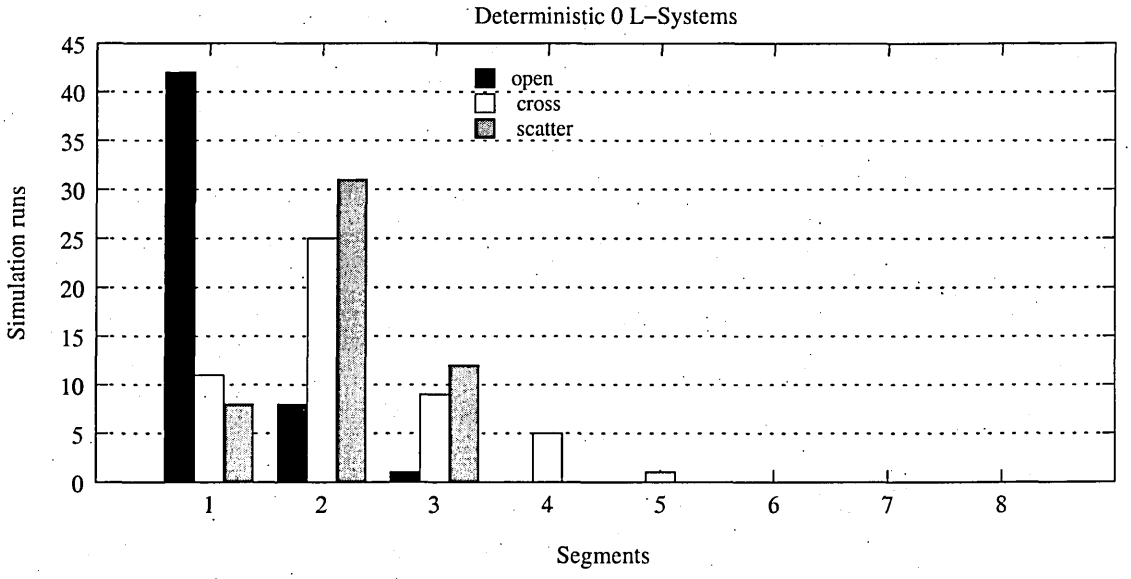


Figure 5.16: Distribution of number of successful simulations for D0L-Systems in each arena arrangement

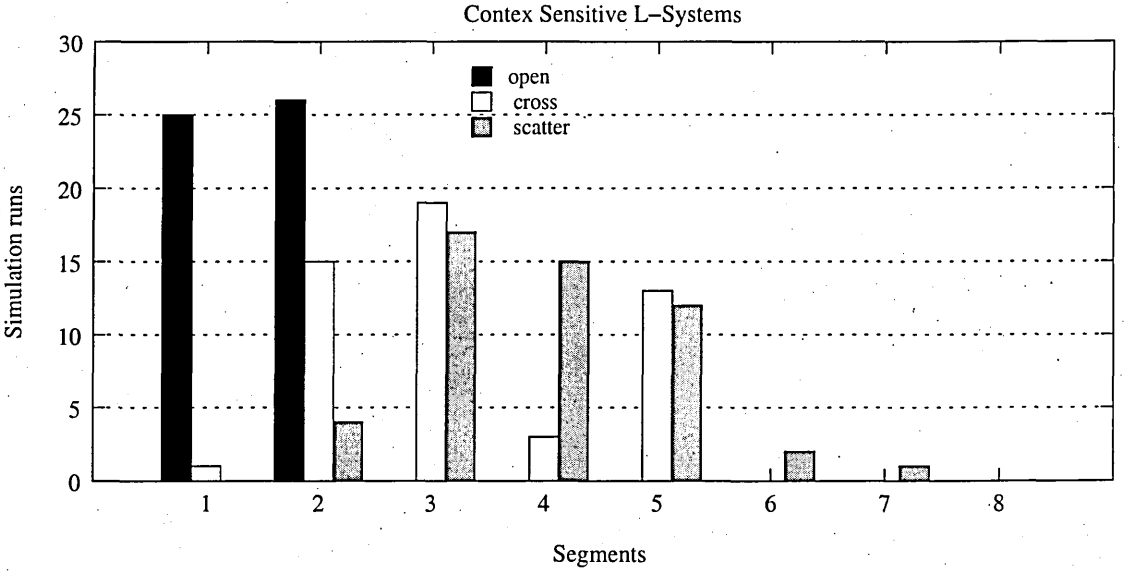


Figure 5.17: Distribution of number of simulations for CSL-Systems in each arena arrangement

Table 5.2: Distribution of number of simulations for each model of L-Systems in each arena arrangement.

	Open arena		Cross arena		Scatter arena	
Segments	DOL	CSL	DOL	CSL	DOL	CSL
1	42	25	11	1	8	-
2	8	26	25	15	31	4
3	1	-	9	19	12	17
4	-	-	5	3	-	15
5	-	-	1	13	-	12
6	-	-	-	-	-	2
7	-	-	-	-	-	1
8	-	-	-	-	-	-

Table 5.3: Average total number of agents for Deterministic 0L-Systems after 51 simulation runs with regard number of segments.

DOL	Open arena		Cross arena		Scatter arena	
Segments	Average	St. dev.	Average	St. dev.	Average	St. dev.
1	58.1	7.44	64.55	2.5	56	0
2	59.2	5.55	75.04	16.41	64.9	11.76
3	70	0	108.22	28.35	72.58	14.34
4	-	-	118.4	14.15	-	-
5	-	-	141	0	-	-
6	-	-	-	-	-	-
7	-	-	-	-	-	-
8	-	-	-	-	-	-

Table 5.4: Average total number of agents for Context-sensitive L-Systems after 51 simulation runs with regard number of segments

CSL	Open arena		Cross arena		Scatter arena	
Segments	Average	St. dev.	Average	St. dev.	Average	St. dev.
1	60.04	5.63	60	0	-	-
2	69.56	10.24	78.87	11.72	65.5	8.58
3	-	-	90.89	20.11	85.53	12.35
4	-	-	119.33	25.15	111.93	22.8
5	-	-	127.08	18.5	141	26.8
6	-	-	-	-	148	46.67
7	-	-	-	-	160	0
8	-	-	-	-	-	-

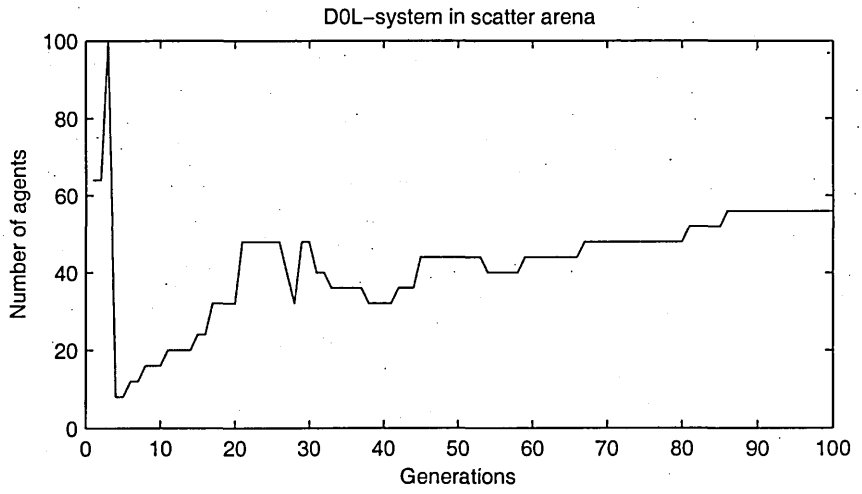


Figure 5.18: Number of agents against generations during the evolutionary process from one of the DOL-System samples in the scatter arena

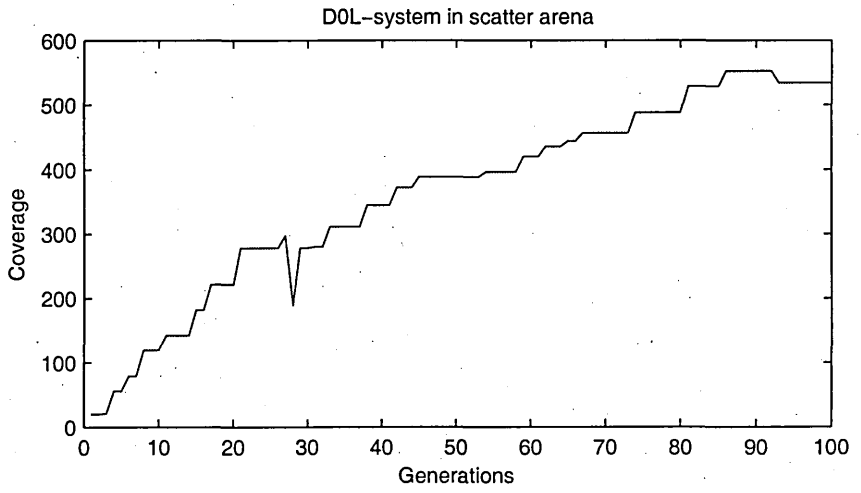


Figure 5.19: Coverage against generations during the evolutionary process from one of the DOL-System samples in the scatter arena

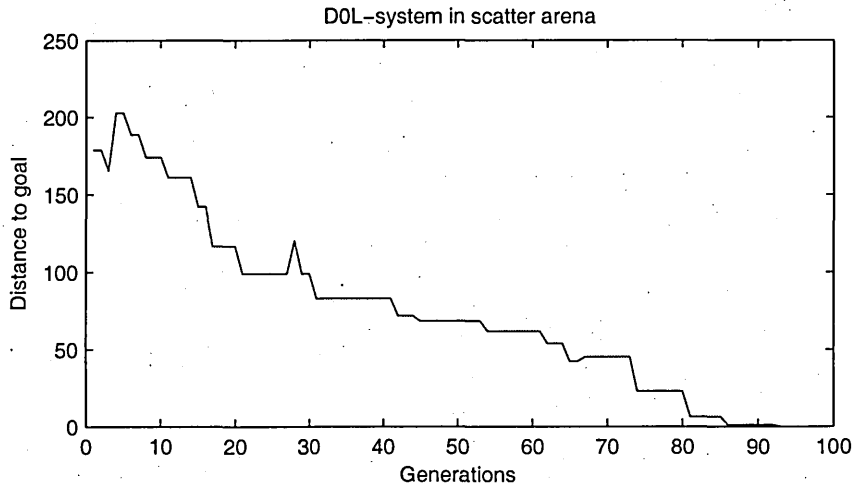


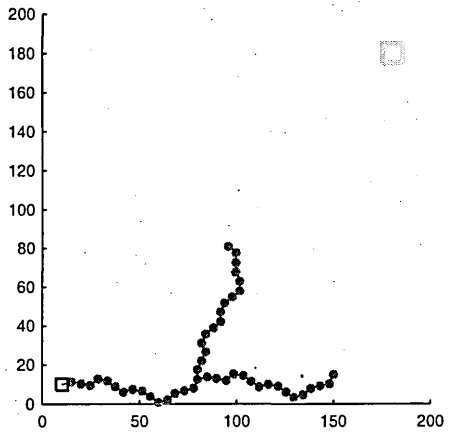
Figure 5.20: Nearest distance to goal against generations during the evolutionary process from one of the DOL-System samples in the scatter arena

The plots in Figure 5.21 are snapshots taken during the evolutionary process of a formed pattern by the D0L-Systems from one of the simulation runs in the open arena. In this instance, Figure 5.21(a)-(e) show the development of the first piece (or segment) of the D0L-System, and Figure 5.21(f) shows the final formed pattern. In Figure 5.21(a)-(c), it is clearly visible that the L-Systems evolved into "Y" shaped pattern. This is due to the the axioms (α) for the L-Systems that consist of the bracket symbols, *[* and *]*, which contribute to the branching structure in the formation. The axioms α, successors, *succ* and the number of iterations for the formations are summarised as follow:

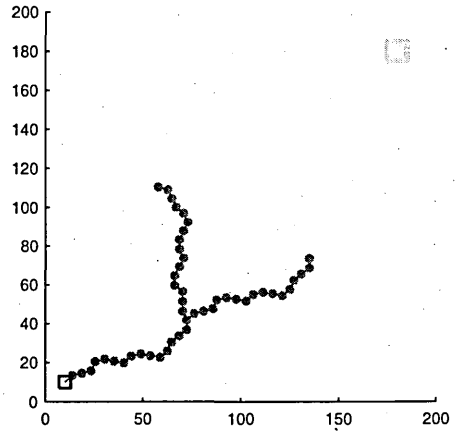
- Figure 5.21(a): *RF[LLLF]FL*, *RFRFFLLF*, 2
- Figure 5.21(b): *F[LLLF]FL*, *RFRFFLLF*, 2
- Figure 5.21(c): *LF[LLF]FL*, *RFRFFFLLF*, 2
- Figure 5.21(d): *LF[LL]FL*, *RFRFFFLLF*, 2
- Figure 5.21(e): *RFLF*, *FFRFFFL*, 2
- Figure 5.21(f): *FLLF*, *LLLLFFL*, 2

In Figure 5.21(d), even though the axiom (*LF[LL]FL*) for the formation contains the bracket symbols for the branching structure, the evolved formation does not have a visible branch. The reason for this is that in order to have a visible branching structure in the formation, at least one *F* symbol needs to reside inside the bracket symbols, as such the axioms for Figure 5.21(a)-(c). Figure 5.21(e) shows the final formation for the first segment at generation 48.

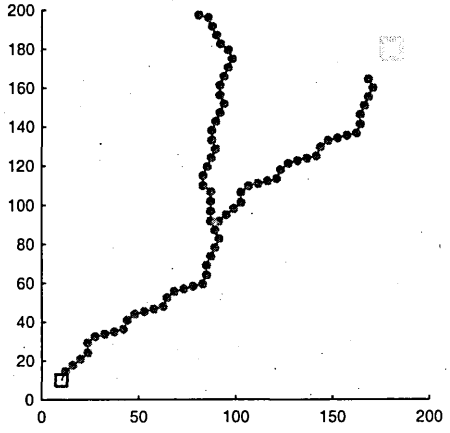
Figure 5.21(f) shows the final formation that reached the goal; in this formation, the second segment of the L-System was grown from the nearest agent to the goal from the first segment of formation. The total number of agents that are required to construct the formation between the *start* and *goal* locations in this instance is 58, consisting of 50 agents in the first segment and 8 agents in the second.



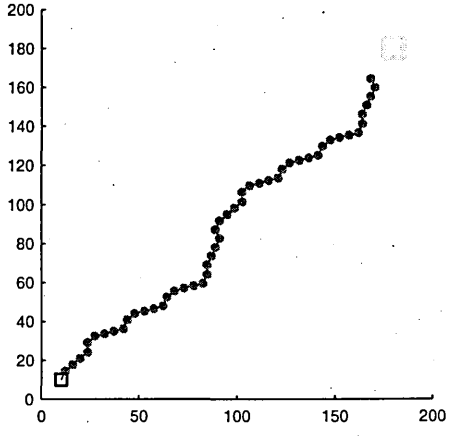
(a) first segment, generation 2



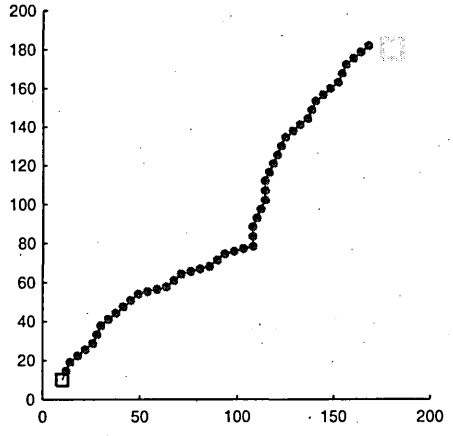
(b) first segment, generation 4



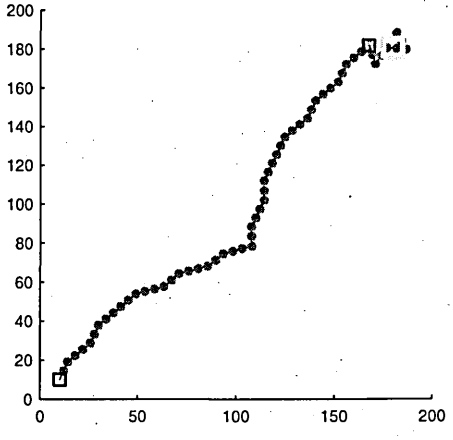
(c) first segment, generation 6



(d) first segment, generation 12



(e) first segment, generation 48



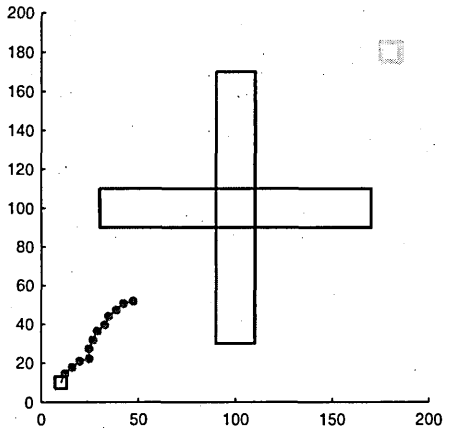
(f) second segment, generation 28

Figure 5.21: Evolving pattern formation of D0L-Systems in the open arena. The fittest L-System for first segment in the: (a) 2nd, (b) 4th,(c) 6th, (d) 12th, (e) 48th generation; second segment in the: (f) 28th generation, of the evolutionary process.

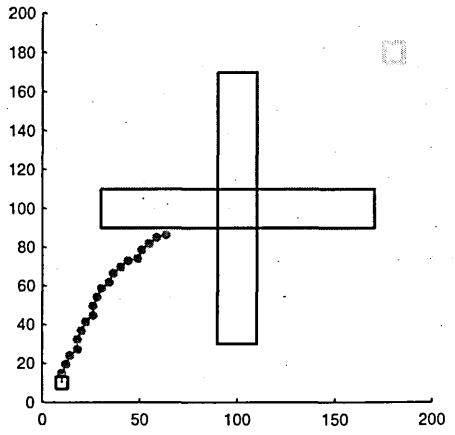
Figure 5.22 is the collection of snapshots showing the evolution of the D0L-Systems in the cross arena from one of the simulation results. Figure 5.22(a)-(d) show the growth of the pattern for the first segment of the pattern in the 5th, 10th, 35th and 70th generation, and Figure 5.22(e)-(f) for the second segment in the 10th and 30th generation respectively. The axioms α, successors *succ*, the number of iteration and the number of agents that made the formations are summarised as follow:

- Figure 5.22(a): *RFLFRF*, *FRFL*, 2, 12
- Figure 5.22(b): *FFFRFFRR*, *FRFL*, 2, 20
- Figure 5.22(c): *LFFFRFRF*, *FRFFL*, 2, 45
- Figure 5.22(d): *LFFFFRRFFRFF*, *FRFL*, 3, 64
- Figure 5.22(e): *R[RFFLL]*, *RRFRF*, 2, 8
- Figure 5.22(f): *R[LFL]RF*, *RRFRF*, 2, 8

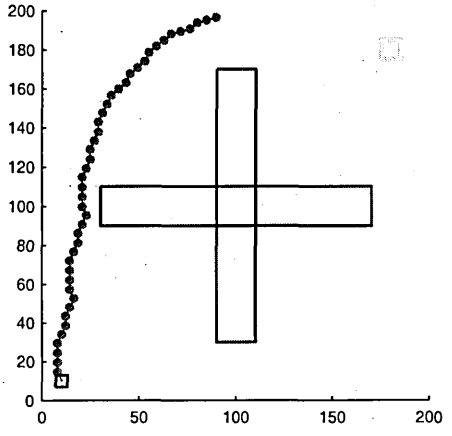
Figure 5.22(d) shows the final formation for the first segment of the L-System. In this instance however, the formation has not reached the target yet, and the second segment needs to be grown. Figure 5.22(e)-(f) show the growth of the second segment. Even though the number of agents and the successor strings in these two are the same, the the evolved formations are made up from different axioms. The total number of agents in this formation is 72, consisting of 64 agents in the first segment and 8 in the second.



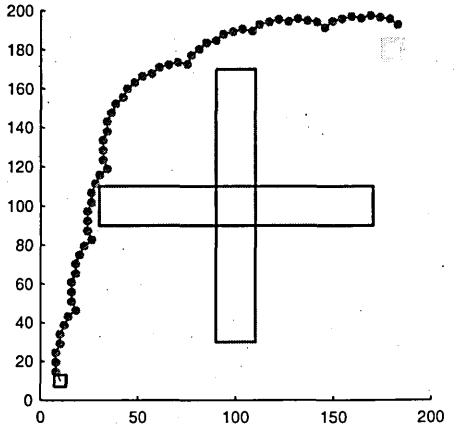
(a) first segment, generation 5



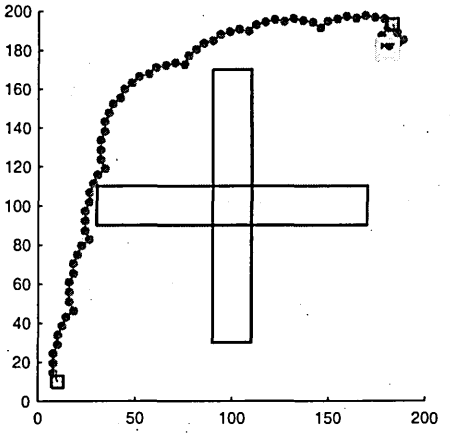
(b) first segment, generation 10



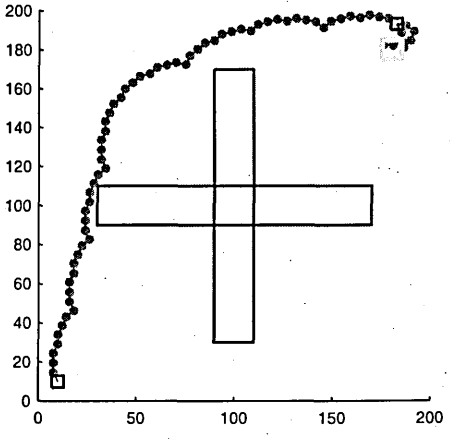
(c) first segment, generation 35



(d) first segment, generation 70



(e) second segment, generation 10



(f) second segment, generation 30

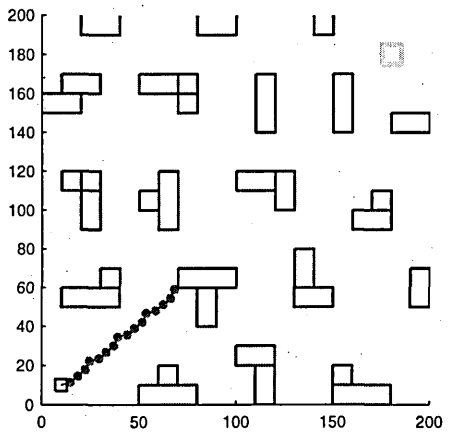
Figure 5.22: Evolving pattern formation of D0L-Systems in the cross arena. The fittest L-System for the first segment at the: (a) 5th,(b) 10th,(c) 30th, (d) 70th generation; second segment at the: (e) 5th,(f) 30thgeneration, of evolutionary process.

Figure 5.23 shows the growth of D0L-Systems in the scatter arena from one of the simulation results. The axioms *α*, successors *succ*, the number of iteration and the number of agents that made the formations are summarised as follows:

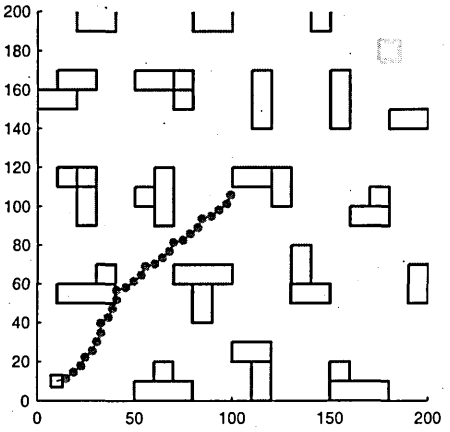
- Figure 5.23(a): *FLFFFFFF*, *FFRR*, 2, 12
- Figure 5.23(b): *RFLFFRFFFF*, *RFLF*, 3, 24
- Figure 5.23(c): *FFFFRF*, *RFLF*, 3, 40
- Figure 5.23(d): *RR[FLFL]*, *RFR[LL[LF]R]FL*, 2, 18
- Figure 5.23(e): *F[LLFFL]*, *FRFLLLFRRR*, 2, 27
- Figure 5.23(f): *FL[F]L*, *FRFLLLFRRFR*, 2, 32

Figure 5.23(a)-(c) show the evolution of the first segment at 5th, 25th and 60th generation, whilst Figure 5.23(d)-(f) show the evolution of the second segment at 5th, 10th and 40th generation respectively. Figure 5.23(d) clearly shows the interesting branching formation which is due to the stacks (and represented by the square bracket symbols) in both the axiom and the successor. Figure 5.23(f) shows the final formation connecting *start* and *goal* location with 2 segments of the L-Systems, in this instance the total number of agents that is able to produce the formation is 72, consisting of 40 agents for the first segment and 32 for the second.

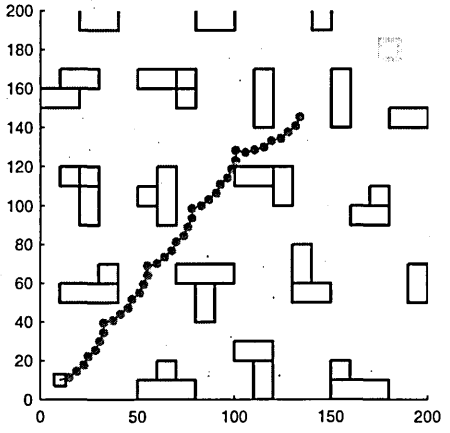
Figure 5.24, Figure 5.25 and Figure 5.26 show the evolution of the CSL-Systems in the open, cross and scatter arena respectively. Figure 5.24(a)-(d) show the evolution of the first segment of the CSL-Systems at 2nd, 6th, 8th and 12th generation respectively. From the snapshots, it is obvious in the growth of the CSL-Systems that as the number of generations increases the nearer the closest agent to the *goal* location becomes. Figure 5.24(e)-(f) depict the evolution of the second segment of the formation with the number of agents at the 2nd generation is 17, and decrease to 12 at generation 32. Figure 5.24(f) shows the final formation which connects the *start* and *goal* location that is made up of 2 segments of CSL-Systems; with the number of agents is 50 for the first segment and 12 for the second segment, making the total of 72 number of agents in the formation.



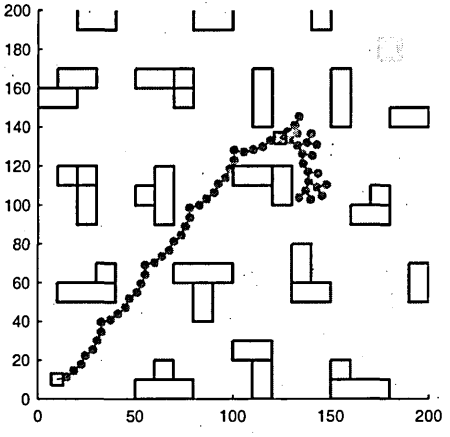
(a) first segment, generation 5



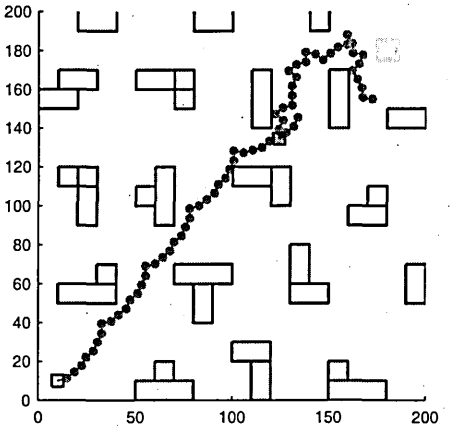
(b) first segment, generation 25



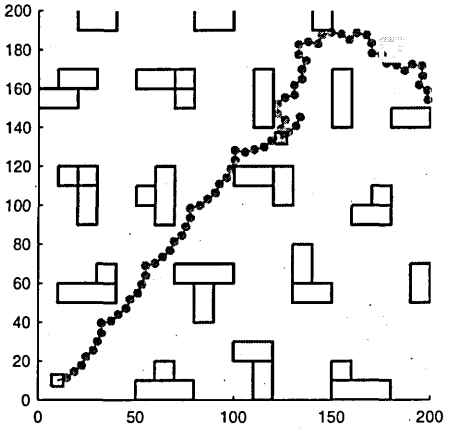
(c) first segment, generation 60



(d) second segment, generation 5

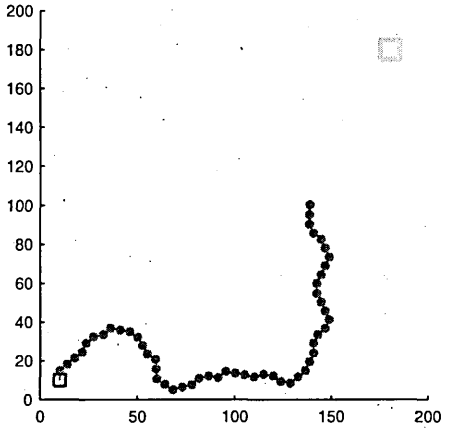


(e) second segment, generation 10

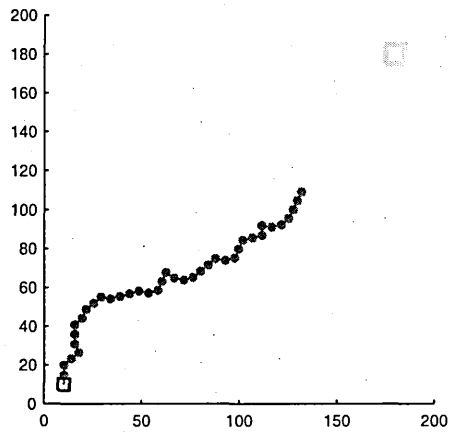


(f) second segment, generation 40

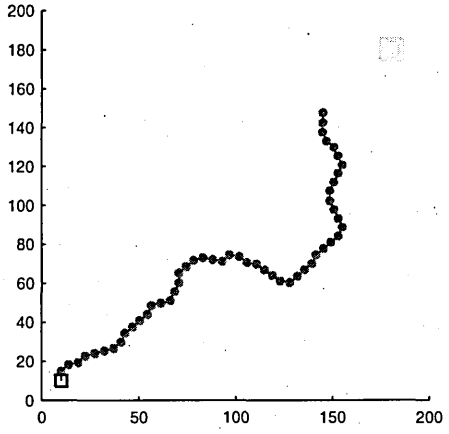
Figure 5.23: Evolving pattern formation of D0L-Systems in the scatter arena. The fittest L-System for first segment in the: (a) 10th, (b) 25th,(c) 60th generation; second segment in the: (d) 5th, (e) 10th, (f) 35th generation, of evolutionary process.



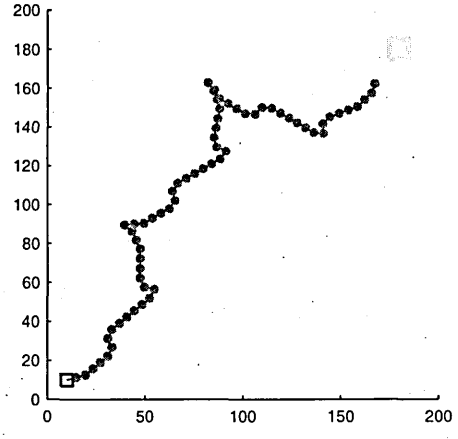
(a) first segment, generation 2



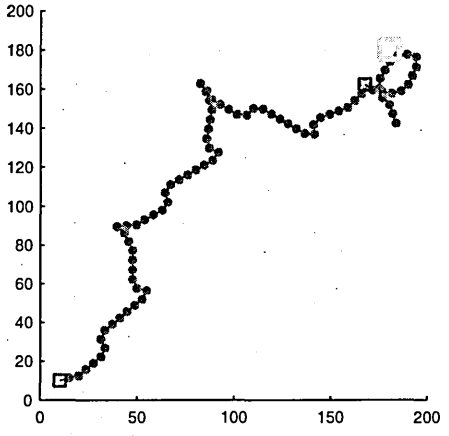
(b) first segment, generation 6



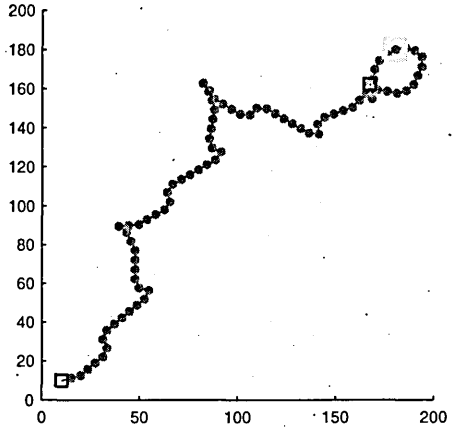
(c) first segment, generation 8



(d) first segment, generation 12

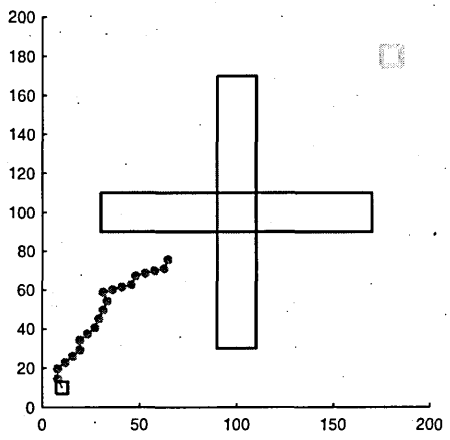


(e) second segment, generation 2

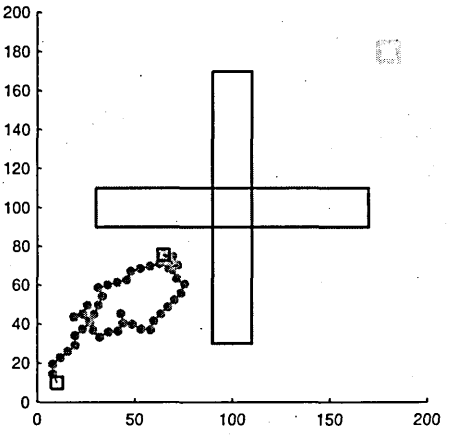


(f) second segment, generation 32

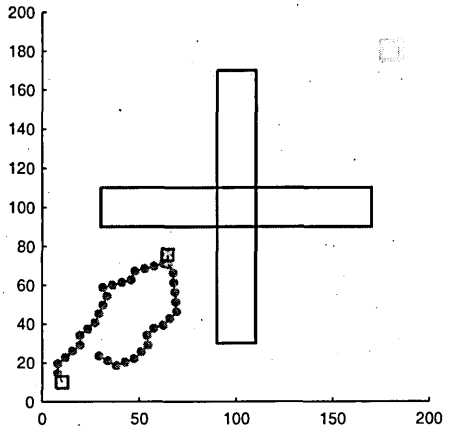
Figure 5.24: Evolving pattern formation of CSL-Systems in the open arena. The fittest L-System for first segment in the: (a) 2nd, (b) 6th,(c) 8th, (d) 12th generation; second segment in the: (e) 2nd, (f) 32nd generation, of the evolutionary process.



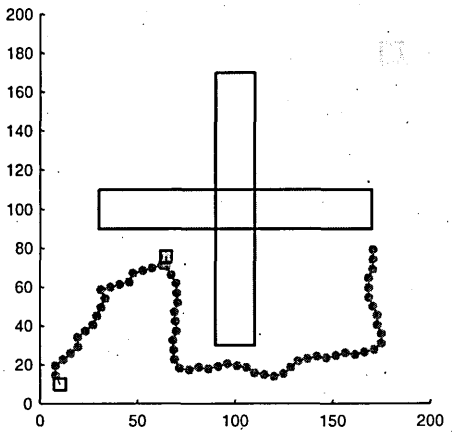
(a) first segment, generation 25



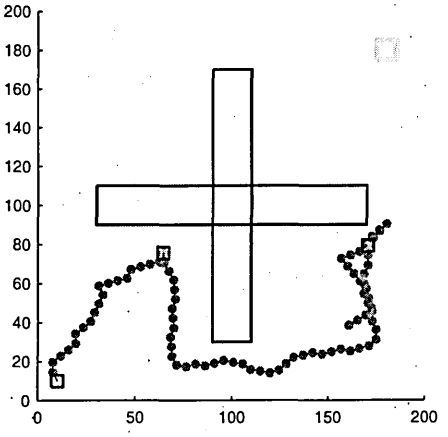
(b) second segment, generation 5



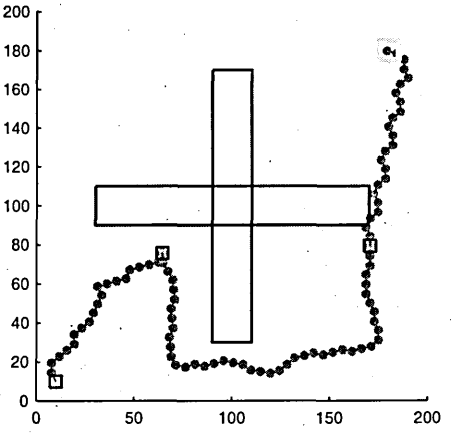
(c) second segment, generation 15



(d) second segment, generation 50

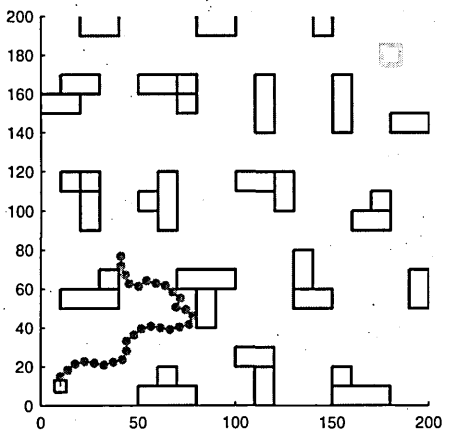


(e) third segment, generation 5

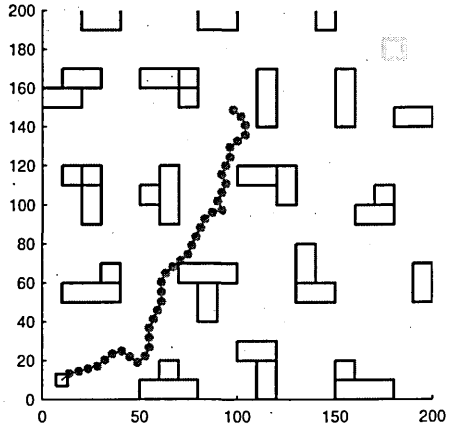


(f) third segment, generation 30

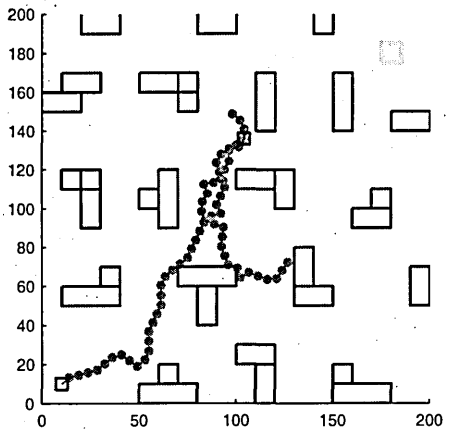
Figure 5.25: Evolving pattern formation of CSL-Systems in the cross arena. The fittest L-System for the first segment at the: (a) 25th generation; second segment at the: (b)5th generation, (c) 15th generation, (d) 50th generation; third segment: (e) 5th generation, (f) 30thgeneration, of evolutionary process.



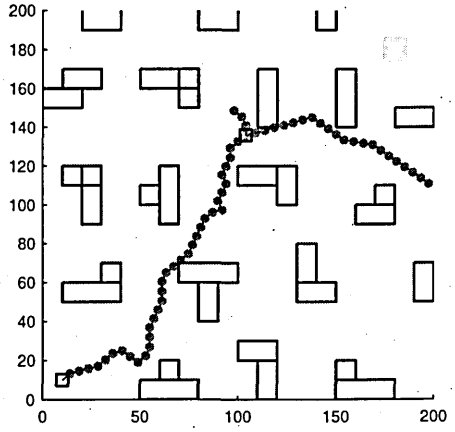
(a) first segment, generation 5



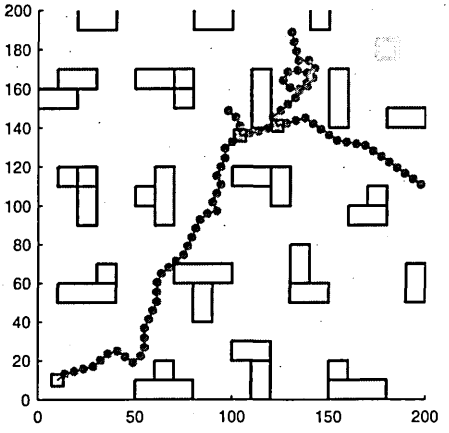
(b) first segment, generation 25



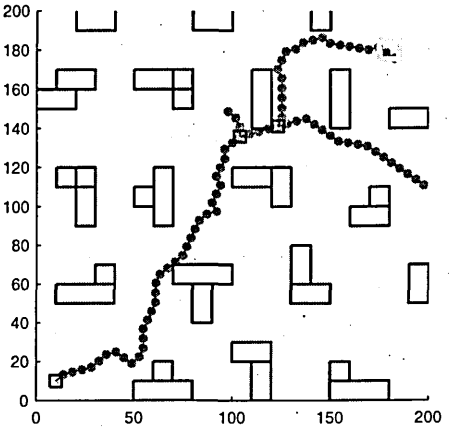
(c) second segment, generation 10



(d) second segment, generation 30



(e) third segment, generation 15



(f) third segment, generation 45

Figure 5.26: Evolving pattern formation of CSL-Systems in the scatter arena. The fittest L-System for first segment in the: (a) 5th, (b) 10th generation; second segment in: (c) 10th, (d) 30th generation; third segment in the; (e) 15th, (f) 45th generation, of evolutionary process.

Figure 5.25(a) depicts the formation of the first segment of the CSL-System at generation 25 with 20 agents. Figure 5.25(b)-(d) show the growth for the second segments; where in Figure 5.25(b) and (c), the formations seem to make a "U-turn" towards the *start* location. Figure 5.25(d) shows the last generation for the second segment with 44 agents required to construct the formation. Figure 5.25(e)-(f) show the evolution of the third segment at generations 5 and 30 respectively. Figure 5.25(f) shows the final formation, in this instance the formation is made up of three segments of CSL-Systems with total number of agents of 88, from which 20 agents are required for the first segment, 44 agents for the second segment and the remainder are for the third segment.

Figure 5.26(a)-(b) show the formation of the first segment at generation of 5 and 25, which requires 30 and 40 agents to form respectively. Figure 5.26(c)-(d) depict the formation growth for the second segment. Figure 5.26(c) shows that the evolved formation is moves further away from the *goal* location, whilst in Figure 5.26(d) the formed pattern seems to grow towards the *goal* location. Figure 5.26(e)-(f) show the evolution of the third segment at 15^{th} and 45^{th} generation. Figure 5.26(f) is the final formation that connects the *start* and *goal* location. The total number of agents in the formation is 81, from which 40 is required for the first segment, 21 for the second segment and 20 for the third segment.

Table 5.5 is the tabulated data for the average, standard deviation, median and minimum number of agents for D0L-Systems and CSL-Systems in each arena respectively. From the data, for overall performance which is based on the average total number of agents, the D0L-Systems seem to outperform the CSL-Systems in every arena arrangement. The open arena ends up with the smallest number of agents, followed by the scatter arena and the cross arena. The minimum number of agents for each arena arrangement between D0L- and CSL-Systems does not differ significantly. For the open arena, the minimum number of agents recorded for D0L- and CSL-Systems are 50 and 52; for the scatter arena they are 53 and 56; and for the cross arena both require 60 agents respectively.

5.4.3 Comparison with RGT and A* search algorithms

Comparative studies of the formed formation between the two locations of evolutionary L-Systems with RGT (Random Growing Tree) and A* search algorithms have also been carried out in this Chapter. For the theoretical background on RGT and A* search algorithms, readers are advised to refer to Chapter 2, where background descriptions and technical aspects of the algorithms are mentioned.

A note that the A* algorithm is a path finding algorithm unlike the evolutionary L-System algorithm proposed in this chapter which is essentially a pattern formation algorithm. However the final arrangement of both algorithms can be compared.

The comparison that has been done is based on the total number of agents that are needed to form an arrangement connecting *start* and *goal* locations. The simulations for both RGT and A* search have been been carried out using the NetLogo (Wilensky 1999) simulation tool.

For the A* search models, 4-directional search and 8 directional search have been used. The search begins at the *start* location, *i.e.* near the bottom left corner in the Figure 5.15 (page 158), and ends when any of the A* node reaches the *goal* location, *i.e.* the red box near the top right corner in the Figure 5.15.

For RGT models, agents with nonholonomic motion and having a 7 unit perspective range have been used. The 7 unit perspective range seems to be reasonable as the separations between the centre of agents is set to be 5. In the simulations, to avoid over crowding in the arena, the maximum number of agents was set to 250. This figure is acceptable due to the size of the arena being 200 by 200. At the beginning of the simulation, agents are placed randomly in the arena. Agents are then allowed to wander in the arena in search of the two locations (*start* and *goal*) and will finally arrange into a formation connecting the two locations by obeying the rules of the RGT algorithm.

The following results are obtained using the same arena arrangements (open, cross and scatter)

as previously used. As before, 51 runs are made for each algorithm (4-directional A* search, 8-directional A* search and RGT) against each arena arrangement. In the A* search algorithms, the total number of agents that are needed to form the path along the route computed by the A* algorithm is defined by the route length divided by 5. This is due to the fact that in the evolutionarily L-Systems method, the symbol *F* represents 5 units of displacement. Figure 5.27 and Figure 5.28 show the snapshots from one of the simulation run for RGT, A* 4-directional and A* 8-directional methodologies.

Table 5.6 shows the tabulated data for average total number of agents that are needed to form the arrangement between the start and goal locations, its standard deviations, medians and minimums for RGT, 4-directional A* search and 8-directional A* search respectively. From the data (Table 5.6), for overall performance which is based on the average total number of agents, 8-directional A* search outperform others in every arena arrangement. The number of agents in the open arena is the smallest (for each method) compared to any other arena.

Table 5.5: Average, median and minimum total number of agents for Deterministic 0L- and Context-sensitive L-Systems after 51 simulation runs in each arena arrangement.

	Open arena		Scatter arena		Cross arena	
	D0L	CSL	D0L	CSL	D0L	CSL
Average	58.51	64.71	65.31	108.69	84.18	97.65
Std. Deviation	7.25	9.45	12.44	33.17	26.22	26.39
Median	56	63	60	109	72	90
Minimum	50	52	53	56	60	60

Table 5.6: Average, median and minimum total number of agents for Random Growing Tree (RGT), A search with 4 directions (A* (4)) and A* search with 8 directions (A* (8)) methods over 51 simulation runs in each arena arrangement.*

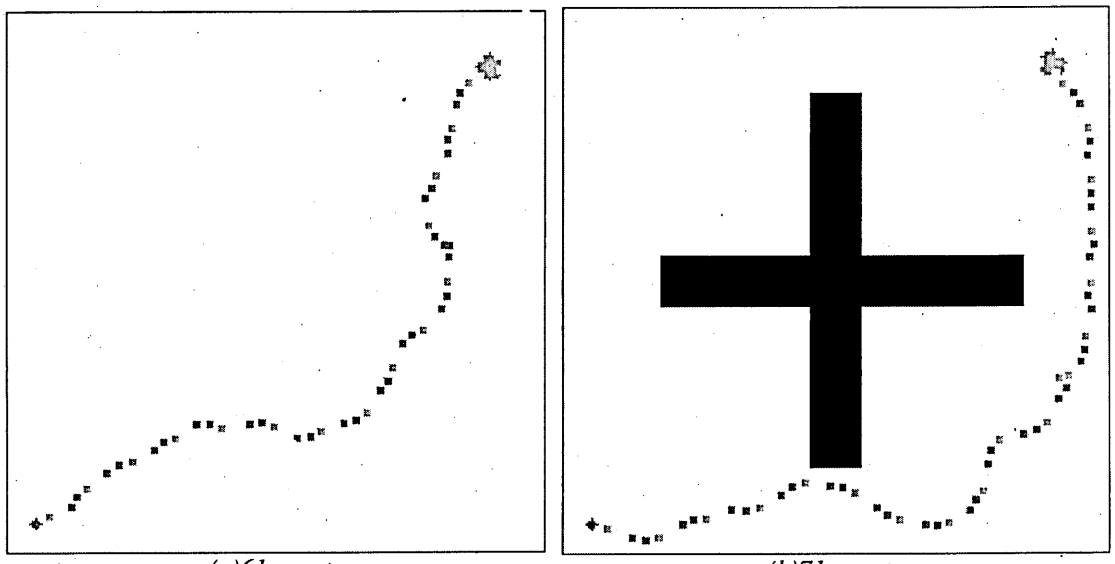
	Open arena			Scatter arena			Cross arena		
	RGT	A* (4)	A* (8)	RGT	A* (4)	A* (8)	RGT	A* (4)	A* (8)
Average	69.45	68	48	80.1	72	53	79.41	88	61
Std. Deviation	8.6	0	0	17.09	0	0	10.45	0	0
Median	69	68	48	75	72	53	75	88	61
Minimum	54	68	48	59	72	53	66	88	61

From Table 5.5 and Table 5.6, shows the overall performances over 51 simulation runs based on the average number of agents. It shows that in the open arena the evolutionary L-Systems

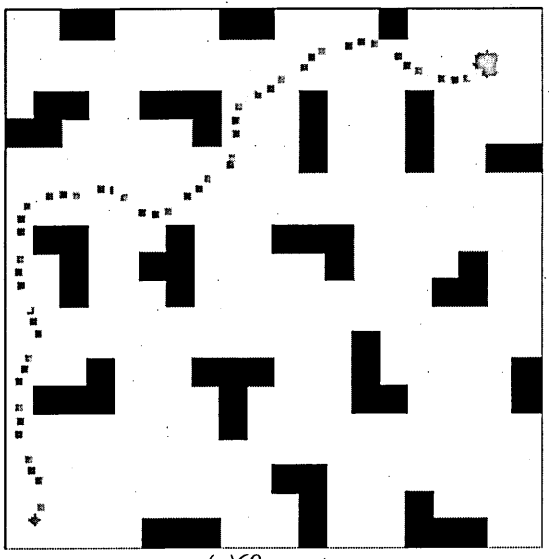
methods performs better than RGT and 4-directional A* search; in the scatter arena, the D0L-Systems performs better than RGT and 4-directional A* search, whilst the CSL-Systems performs the worst. Finally in the cross arena both D0L- and CSL-Systems perform worse than other methods.

Focus on the last row of Table 5.5 and Table 5.6, *i.e.* the smallest number of agents or the best result recorded over 51 simulation runs that is needed to form a connection between the *start* and *goal* locations. The smallest number of agents is considered to be best as it contributes to the shortest formation between the two locations. From the results, it is shown that the evolutionary L-Systems methods are able to perform better than RGT and 4-directional A* search techniques. With regards to the 8-directional A* search technique, there is little difference between it and the best results from the evolutionary L-Systems. For the open arena, the D0L- and CSL-System needs 50 and 52 agents, while 8-directional A* search needs 48. For the scatter arena, the D0L-Systems and 8-directional A* are on a par with 53 agents, whilst the CSL-Systems needs 56. Finally for the cross arena, D0L- and CSL-Systems perform better with 60 agents than the 8-directional A* search which needs 61 agents to connect between the *start* and *goal* locations.

Figure 5.29(a)-(c) show the snapshots of the best result for the D0L- and Figure 5.29(d)-(f) for CSL-Systems in the open, cross and scatter arena respectively. In this Chapter, what we define as the best result is the bridging formation between *start* and *goal* location with the least number of agents required. For the best formation, it does not necessarily comes from one-piece or one-segment of the L-System. Figure 5.29(b),(c) and (f) show that, the formed arrangements are made up of two segments of the L-Systems.

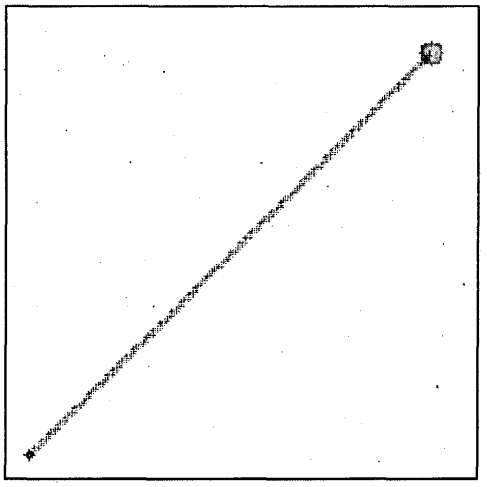


(a)61 agents *(b)71 agents*

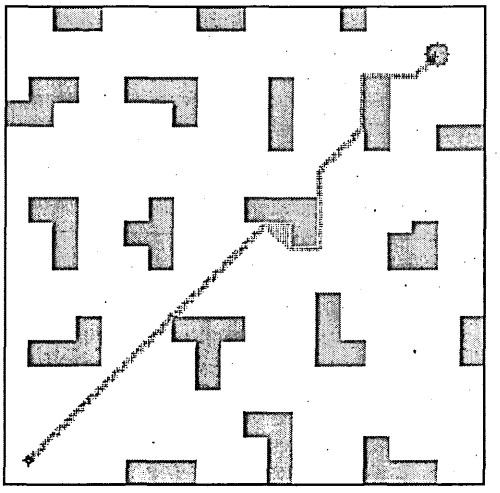


(c)69 agents

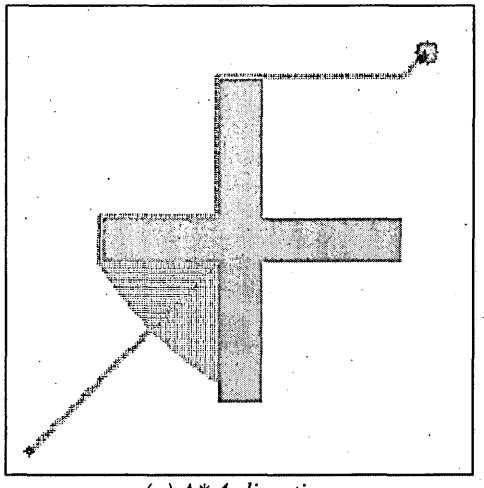
Figure 5.27: Example of formed pattern using RGT method. Small coloured squares represent agents position. (a) in open arena with total of 61 agents. (b) in cross arena with 71 agents. (c) in scatter arena with 69 agents.



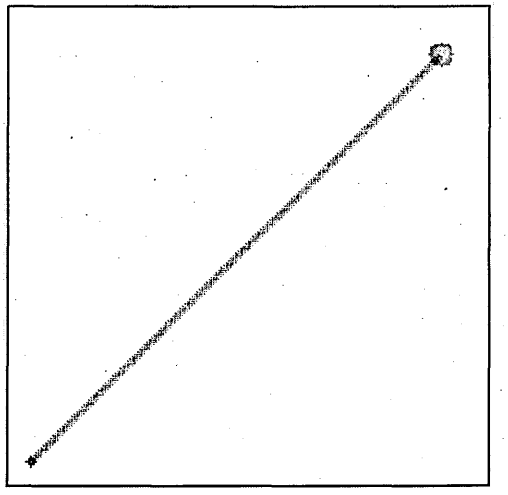
(a) A* 4-direction



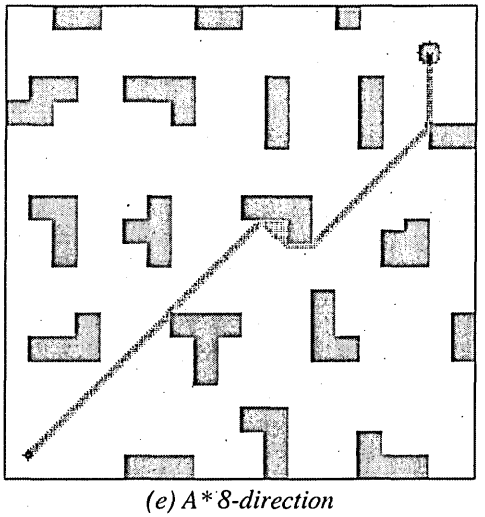
(b) A* 4-direction



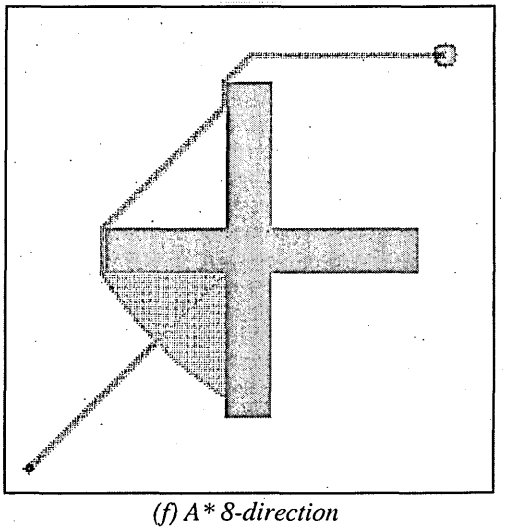
(c) A* 4-direction



(d) A* 8-direction

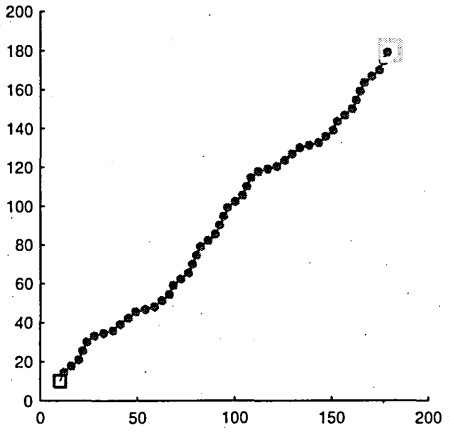


(e) A* 8-direction

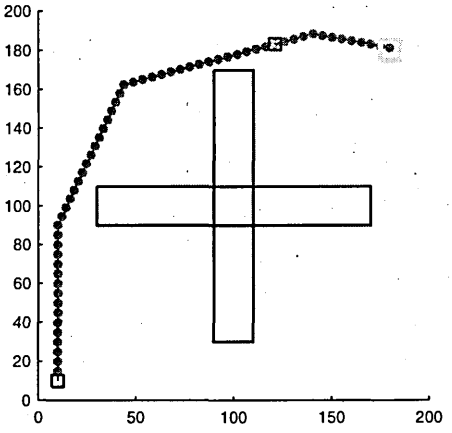


(f) A* 8-direction

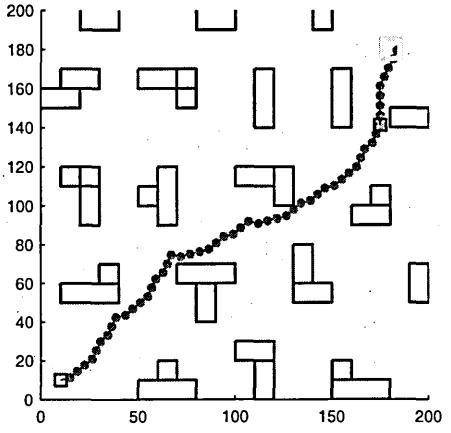
Figure 5.28: Formed pattern using (a)-(c) A 4-direction and (d)-(e) 8-direction methods, in the open (a)(d), scatter (b)(e), and cross (c)(f) arena respectively.*



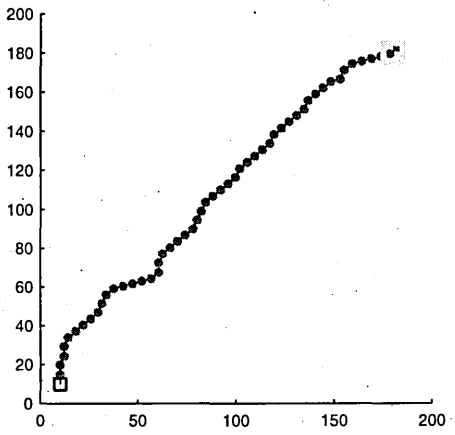
(a) D0L-open arena



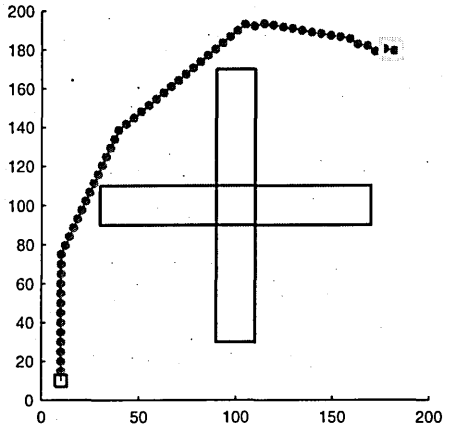
(b) D0L-cross arena



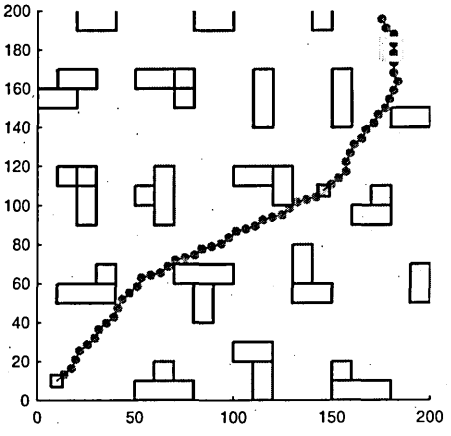
(b) D0L- scatter arena



(d) CSL- open arena



(e) CSL- cross arena



(f) CSL- scatter arena

Figure 5.29: Evolved L-Systems robot swarms formation. Each figure show the "best" generated results for: D0L-Systems in (a) open, (b) cross, (c) scatter arena; and for CSL-Systems in (d) open, (e) cross, (f) scatter arena, respectively.

5.5 Summary Remarks

Due to the limited amount of communication bandwidth in swarm systems, there is a need to design algorithms that require minimum transfer of information. This Chapter has introduced a new and original method for organised formation along a path in large scale multi-agent systems which can also be used as a path planning algorithm. The method however, requires the pre-evolution of patterns that are represented by L-Systems. By developing this L-Systems method, complex pattern formation information can be stored as short bitstrings that can be communicated to neighbouring agents, thus fulfilling the requirement for minimum communication. Through the use of L-Systems, complex formations need not be explicitly encoded. Instead, these formations can be evolved by specifying objectives in the form of fitness functions that are fed into a evolutionary engine.

From the tabulated data in Table 5.5 and Table 5.6 (page 176), the overall results based on the average of the total number of agents that are needed to form the formation along the path, do not favour L-Systems. However, based on the least number of agents needed to form the arrangements, and by altering the stop condition of the evolutionary process of the L-Systems, the overall results can be improved. The alteration in this case can be done by:

- increasing the maximum number of generations, and / or
- increasing the limit for stagnancy.

The aim of this Chapter was to investigate an alternative way for swarm agents to form an arrangement along a path between two locations. In order to be able to form the formations, agents are required to have the ability to interpret short strings of the L-Systems that form the basic DNA of the formation.

The goal in this Chapter was to achieve interesting and complex pattern formations of robot swarms by evolving L-Systems. What makes the L-System attractive is the way the representation of pattern takes place. Consider the formation of the evolved D0L-System in the open arena as in Figure 5.29(a) (page 180). Such pattern formation can be presented in the

Logo-style string format as follows:

- *RRRFFLFFFRFFLFFFFFLFFFRFFLFFFRFFLFFFRRFFLFFFRFFLFFFFFLF FFRFFLFFFRFFLFFF*

However, the same pattern formation can be represented by the L-Systems with a shorter string in the format of "*axiom* > *successor* > *number of rewriting operation*" as follows:

- *RFF* > *RFFLFFF* > 2

where the ">" symbol is use to separate between different parts of the L-Systems parameters. Due to the Logo-style format, these movement can be fed directly to the robots.

The technique in this Chapter was mainly developed for the use of forming a formation along the path between locations. As already mentioned previously, L-Systems use Logo-style format to represent the formation, thus the developed technique in this Chapter can also be used as a new path planning algorithms.

Furthermore, the ability to represent branching structure or pattern makes L-Systems more appealing. This ability is particularly useful when formations of agents in connecting three or more locations are needed. For example, the result in Figure 5.21(c) (page 166) which shows the branching structure of the pattern. Assume that agents are needed to form a bridging formation connecting three locations, and the three locations are in fact at the edge of every branch of Figure 5.21(c) formation. By using the proposed technique, the representation of the pattern (as in Figure 5.21(c)) which uses L-Systems and Logo-style format only takes minimal string length, as follows:

- *LF[LLF]FL* > *RFRFFFLLF* > 2

The results on the different arena arrangements provided the basis for the study of the formed patterns by the evolutionary L-Systems. From the two models (D0L- and CSL-Systems) simulated, it was obvious that the D0L-Systems model produced better results with the least number of agents to form the bridging formation between the *start* and the *goal* locations. Furthermore, representing the D0L-Systems can be done by only using three sets of strings

(axiom, successor, number of rewriting) compared to CSL-Systems which needed eight (axiom, 2 contexts, 4 successors and a number of rewriting). However, the patterns formed by the D0L-System models lacks what we shall term complexity compared to the CSL-Systems model. In this case, the patterns from the D0L-Systems are somewhat symmetrical. But in CSL-Systems model, the patterns appear to be more random. For this reason, it is believed that to some extent the evolutionary CSL-Systems model will outperform the D0L-Systems model given the right conditions. However this could be conducted as future work, as this thesis is mainly aimed at laying a new paradigm for the topic of formation.

Chapter 6 Conclusions and Future Work

6.1 Overview

The research presented in this thesis provides an important early contribution to researchers currently working on various different themes that fall into the domain of applied swarming, namely swarm engineering, swarm robotics, swarm intelligence, multi-agent systems, and so on and so forth. In general, work in these fields refers to approaches of developing swarms of relatively simple and independent agents which are capable of completing specific global tasks, either through task allocation or emergent behaviour.

Swarm robotics has a strong link with multi-agent (robotics) systems, where problem solving is done at a macroscopic level. In one sense, designing microscopic rule sets for homogenous agents to achieve macroscopic goal(s) may seem to be a simple task, as all the agents will have the same rules and conditions. One could strive to design these rules and conditions by hand. Moreover, the generation of an analytical solution to the problem might not be required[2], although in some cases analytical and exact solutions are a must. Any behaviour (of agents) which satisfies the macroscopic goal can be thought of as a solution to the problem.

The field of multi-agent mobile systems is still young, hence the current lack of physical swarms. Many current problems on swarming have been addressed by analysing and understanding biological swarms, and many problems on swarm robotics have been solved in the wider context of artificial intelligence and robotics. In both respects, ideas are borrowed and adapted. However there remain many more issues that are yet to be solved. Such issues include

2 This is based on the author's observation on many other works of research

the issue of how to control agents which have very limited memory and ability to form interesting patterns; the issue of bridging formation amongst agents connecting multiple objects; and the issue of flocking behaviours with the existence of an attractor in the arena.

In this thesis, three different pieces of research that are related to centralised and decentralised pattern formation have been studied. The three systems were different in the inherent nature of the problem and in the type of solution. The first piece of research is based on the state based model (Chapter 3). In this research, homogenous agents with very little memory, limited sensing capabilities and processing power were designed and modelled for two types of swarm behaviours, *i.e.* line formation and cluster formation, using the well-known Finite State Machine approach.

The second piece of research addresses the problem of collective movement modelling (Chapter 4). In this work, the macroscopic behaviour of swarm agents in the presence of an attractant (artificial potential field) is studied.

The third piece of research (Chapter 5) studies complex formation of agents in a task that requires the bridging connection of two locations. In this work, it has been shown that propagatable patterns can be represented by using L-Systems, provided each robot has the ability to interpret short strings of L-Systems that form the basic DNA of the formation.

6.2 Original Contributions to Knowledge

The contributions to knowledge of the three pieces of research above is thus presented.

6.2.1 State based models

The goals of state based models (Chapter 3) were:

- to design relatively simple homogenous swarm agents with very little memory and limited sensing capabilities, and
- to devise algorithms for the agents so that agents will self-organise into patterns in a

decentralised manner.

The tasks that have been chosen in this work are that the agents have to perform line and cluster formation. As the agents have very little memory, limited sensing capabilities and processing power the agents themselves do not have any knowledge of the arena and how many other agents are present in the environment. Due to the constraints, an FSM approach has been chosen and applied. Using FSM as an approach is not a new idea, after all behaviour-based systems have used FSM as their backbone.

There are many similar works have been reported on distributed pattern formation control algorithms of robot swarms. However, these agents have capabilities that are vastly more complex than the requirement of simple agents in swarm systems. For example, in Avrutin *et al.* (2007), Payton *et al.* (2004), Nouyan *et al.* (2006), Freeman *et al.* (2006), Desai (2002), Fierro & Das (2002), Kaminka & Glick (2006), Pavone & Frazzoli (2007), *etc.* agents require a communication module; in Yang *et al.* (2007), Desai (2002), Takahashi (2004), Mastellone *et al.* (2007), *etc.* agents need a large amount of memory and processing power for complex calculations; in Das *et al.* (2002) agents require vision based sensors.

In this work, each agent is fitted with a ring of eight equally spaced infrared transmitter-receiver pairs. These infrared pairs are merely used for signalling and obstacle detection rather than full-blown communication.

The main contributions of this research are:

- decentralised line formation algorithm and
- cluster formation algorithm

This is achieved by alternatively switching on and off a combination of transmitters and sensors of relatively simple agents which have very little memory, limited sensing capabilities and processing power.

The line formation algorithm is one in which all agents possesses the same control algorithm. In this algorithm, the agents wander randomly in the arena until it stumbles upon another agent that it can follow. This process promotes autocatalytic behaviour, where in the end, agents would likely end up forming a single long line.

In the cluster formation algorithm, there are two different agent control algorithms. One is for the single attractor agent and another is for the searcher agents. In this algorithm, the attractor agent will wander randomly in the arena, whilst the searchers will look for the attractor and once found, the searchers will follow the attractor.

The control algorithm for searcher agents (in cluster formation) and for agents in line formation algorithm are very similar. They only differ in the on-off configuration of the infra-red transmitters and the receivers.

6.2.2 Collective movements model

The aim of the research on collective movement models (Chapter 4) was to analyse the aggregation behaviour of a large number of agents in a swarm that follows the swarm robotics control paradigm, *i.e.* Reynolds' flocking rules (1987), in the existence of an attractor field.

Within this research, Wilensky's (1999) flocking algorithm has been extended and several individual behaviours have been selected in terms of single-agent movement models. Three different microscopic behaviours have been modelled and a study of the system at a macroscopic level have been conducted. The difference between the microscopic behaviours is the maximum turning angle of each agent. Based on the observation of the movement model, the three behaviours have been labelled as fish-like, mosquito-like and firefly-like.

To summarise, the contributions of this research are:

- Exploration of how flocking agents behave in the existence of an APF.
- The analysis of each movement behaviour in the existence of an APF.

Based on the performances and observation of the movement models, the following conclusion is drawn:

- Teams of collective moving agents with a smaller maximum turning angle are more effective in finding targets than the larger maximum turning angle. Collectively moving agents with a larger maximum turning angle tend to stay close to each other regardless of the APF.

6.2.3 L-Systems for formation tasks

The main objectives of the evolutionary L-System models for formations tasks, as presented in Chapter 5, was to develop a novel theory to support swarm agents for complex pattern formation, where swarm agents are able to form complex patterns between two locations.

In this thesis it is proposed that for more complex pattern formations of swarm agents, the level of agent complexity should be marginally increased. It is thus proposed that to be able to form complex patterns, L-Systems offer one solution, with the assumption that agents will be able to interpret the short L-Systems bit strings.

The tasks that we have chosen are that the agents have to connects two locations in three different arenas in an organised formation. The L-Systems are then generated and evolved by an evolutionary engine that finds for a solution (which is the formation between the two locations). This thesis claims first use of L-systems in the swarm robotics domain.

To summarise, the contributions of this work are:

- the proposal of a pattern construction methodology for swarm robots using L-Systems.
- the proposal of a representation methodology for swarm robot using L-Systems.
- the provision of an empirical study for the use of evolutionary L-Systems for pattern formation in swarm robots.
- the provision of a comparative study of evolutionary L-Systems with other methods (RGT and A* search algorithm).
- explore the approach of a new path planning algorithm.

6.3 Recommendation for Future Works

A number of avenues are available for extending the research in this thesis. They are as follows.

6.3.1 State based models

Currently, the algorithms presented in the research of Chapter 3 are solely based on behaviour-based systems which mimic the line formed by ants and does not deal with any specialised "knowledge". However, in order to deal with more complex behaviour, *i.e.* cooperation amongst agents, each agent has to be aware of its current situation. For that reason, some kind of "knowledge" representation system should be added to each agent. Gershenson (2002) shows how a behaviour-based system is able to abstract knowledge from its environment and exploit this knowledge for performing within its environment by introducing behaviour-based knowledge systems (BBKS). One approach is by using Hidden Markov Models (HMM) (Rabiner & Juang 1996).

In order for each of the agents to sense and understand the world around itself, another area worth investigating is distributed path planning. With path planning, each agent will have part of a "world map" which collectively represents the world. Algorithms need to be developed to account for gaps in the representation by each agent, and to recover when this information becomes available.

6.3.2 Collective movements model

In the work of Chapter 4, there are many areas that can be investigated. These include:

- an investigation into how the population density in the arena affects the swarms' performance and the convergence rate.
- scenarios with more than one attraction field with varying strengths and the effects this will have on an agent's trajectories and the group behaviour of the swarm.
- the modelling of several types of obstacles and an investigation into the emergent behaviours that obstructions may produce.
- the introduction of a moving *attractor* or several *attractors* to analyse their effects to

the swarm.

6.3.3 L-Systems for formation tasks

Future directions of the L-Systems work in Chapter 5 may include:

- Improving the evolutionary engine and the systems. By improving the evolutionary engine, more complex agent formations in a complex arena could be achieved.
- Exploring formation representation connecting three or more locations.
- Evolving a more complex L-Systems model, such as Parametric L-Systems. In a parametric L-Systems model, the representation of the code is more compact than the simple L-Systems.

6.4 Summary

Swarm robotics is a relatively new field which has been investigated in the last decade, having been triggered by Reynolds' (1987) seminal paper on flocking of boids. There has not yet been a single "real world application" of the swarm agents with real physical embodied agents of everyday tasks. This is due to the fact that the early groundwork is still being laid and many problems and tasks exist that will need to be addressed. This thesis has provided an early example of a global approach to pattern formation of swarm agents. The techniques discussed in this thesis may be extended to a wide variety of possible future swarm applications.

References

1. Abelson, H. and deSessa, A. A. (1982). "Turtle Geometry: The Computer as a Medium for Exploring Mathematics". MIT Press, Cambridge, MA. (ISBN: 0262010631).

2. Antonelli, G., Arrichiello, F. and Chiaverini, S. (2005). "The Null-Space-Based Behavioral Control for Mobile Robots". In Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 2005, Finland), pp. 15-20.

3. Antonelli, G., Arrichiello, F. and Chiaverini, S. (2006). "Experiments of Formation Control with Collisions Avoidance Using the Null-Space-Based Behavioral". In Proceedings of 14th Mediterranean Conference on Control and Automation (MED 2006, Italy), pp. 1-6.

4. Antonelli, G., Arrichiello, F., Chakraborti, S. and Chiaverini, S. (2007). "Experiences of Formation Control of Multi-Robot Systems with the Null-Space-based Behavioral Control". In Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2007, Italy), pp. 1068-1073.

5. Arai, T., Pagello, E. and Parker, L.E. (2002). "Guest Editorial: Advances in Multirobot Systems". IEEE Transactions on Robotics and Automation, 18(5), pp. 655-661.

6. Arkin R.C. (1989). "Motor Schema-Based Mobile Robot Navigation". The International Journal of Robotics Research, 8(4), pp. 92-112.

7. Arkin, R.C. (1998). "Behavior-Based Robotics". MIT Press, Cambridge, MA. (ISBN: 0262011654)

8. Augimeri, A., Folino, G., Forestiero, A. and Spezzano, A. (2006). "A Multidimensional Flocking Algorithm for Clustering Spatial Data". In Proceedings of the 7th Workshop From Objects to Agents (WOA 2006, Italy), pp 16-20.

9. Avrutin, V. *et al.* (2007). "I-SWARM, Deliverable N:6.1, Algorithms and Simulations for Collective Perception". I-SWARM internal report, 22nd January 2007, pp. 31-53.

10. Bahceci, E., Soysal, O. and Sahin, E. (2003). "A Review: Pattern Formation and Adaptation in Multi-Robot Systems". Technical Report *CMU-RI-TR-03-43,* Robotics Institute – *Carnegie Mellon University, Pittsburgh, USA.*

11. Balch, T. and Arkin, R.C. (1998). "Behavior-Based Formation Control for Multirobot Systems". IEEE Transactions on Robotics and Automation, 14(12), pp. 926-939.

12. Balch, T. and Hybinette, M. (2000). "Social Potentials for Scalable Multi-Robot Formations". In Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2000, San Francisco, USA), pp. 73-80.

13. Barnes, D. P., Ghanea-Hercock, R. A., Aylett, R. S. and Coddington, A. M. (1997). "Many Hands Make Light Work? An Investigation into Behaviourally Controlled Co-Operant Autonomous Mobile Robots". In Proceedings of First International Conference. on Autonomous Agents (Marina del Rey), pp. 413-420.

14. Bayazit, O.B., Lien, J.-M. and Amato, N.M. (2002). "Better Group Behaviors in Complex Environments using Global Roadmaps". In Proceedings of the 8th International Conference on Artificial Life (Alife 2002, Sydney, Australia), pp 362-370.

15. Bayazit, O.B., Lien, J.M. and Amato, N.M. (2004). "Swarming Behaviour Using Probabilistic Roadmap Technique". In Proceedings of International Workshop on Swarm Robotics (SAB 2004, Santa Monica, USA, revised selected paper), Lecture Notes in Computer Science 3342, pp. 143-152.

16. Bedau, M.A. (2003). "Artificial Life: Organization, Adaptation and Complexity from the Bottom Up". TRENDS in Cognitive Sciences, 7(11) pp. 505-512.

17. Beni, G. (2005). "From Swarm Intelligence to Swarm Robotics". International Workshop on Swarm Robotics (SAB 2004, Santa Monica, USA, revised selected paper), Lecture Notes in Computer Science 3343, Swarm Robotics, pp. 1-9.

18. Beni G. and Wang J. (1989). "Swarm Intelligence". In Proceedings of the Seventh Annual Meeting of the Robotics Society of Japan (Tokyo, Japan), pp. 425-428.

19. Bonabeau, E., Dorigo, M. and Theraulaz E. (1999). "Swarm Intelligence: From Natural to Artificial Systems". Santa Fe Institute Studies on the Sciences of Complexity, Oxford

Press University. (ISBN: 0195131592).

20. Bonabeau, E., Theraulaz, G., Deneubourg, J.-L., Aron, S., and Camazine, S. (1997). "Self-Organization in Social Insects". TRENDS in Ecology and Evolution, 12, pp. 188-193.

21. Braitenberg, V. (1984). "Vehicle: Experiments in Synthetic Psychology". MIT Press, Cambridge, MA.

22. Brooks, R. (1986). "A Robust Layered Control System for a Mobile Robot". IEEE Journal of Robotics and Automation, 2(1), pp.14-23.

23. Brooks, R.A. (1991a). "Intelligence without Reason". In Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI 1991, Sydney, Australia), pp. 569-595.

24. Brooks, R.A. (1991b). "Intelligence without Representation". Journal of Artificial Intelligence, 47, pp. 139-159.

25. Camazine, S., Doneubourg, J.L., Nigel, R.F., Sneyd, J., Theraulaz, G. and Bonabeau, E. (2001). "Self-Organization in Biological System". Princeton Studies in Complexity, Princeton University Press.

26. Camazine, S. and Sneyd, J. (1991). "A Model of Collective Nectar Source by Honey Bees: Self-Organization Through Simple Rules". Journal of Theoretical Biology, 149, pp. 547-571.

27. Cao, Y., Fukunaga, A. and Kahng, A. (1997). "Cooperative Mobile Robotics: Antecedents and Directions". Autonomous Robots, 4, pp. 1-23.

28. Castle, C.J.E. and Andrew T.C. (2006). "Principles and Concepts of Agent-Based Modelling for Developing Geospatial Simulations". Working Paper Series, Paper 110 – Sep 2006, Centre for Advanced Spatial Analysis, University College London.

29. Chen, Y-Q., Zhuang, Y. and Wang, W. (2007). "A Dynamic Regulation and Scheduling Scheme for Formation Control". ACTA AUTOMATICA SINICA 33(6), pp. 623-634.

30. Cheng, J., Cheng, W. and Nagpal, R. (2005). "Robust and Self-Repairing Formation Control for Swarms of Mobile Agents". In Proceedings of The Twentieth National Conference on Artificial Intelligence (AAAI 2005, Pennsylvania, USA), pp 59-64.

31. Cohen, R. and Peleg, D. (2006). "Local Algorithms for Autonomous Robots Systems". In Proceedings of Structural Information and Communication Complexity Conference (SIROCCO 2006, Chester, UK), Lecture Notes in Computer Science, 4056, pp. 29-43.

32. Couzin, I.D. And Krause, J. (2003). "Self-Organization and Collective Behavior in Vertebrates". Advances in the Study of Behavior 2003, 32, pp. 1-79.

33. Curtis, S.A., Mica, J., Nuth, J., Mar, G., Rilee, M., and Bhat, M. (2000). "ANTS (Autonomous Nano-Technology Swarm): An Artificial Intelligence Approach to Asteroid Belt Resource Exploration". In Proceedings of International Astronomical Federation 51st Congress, International Astronautical Federation, 2000.

34. Cresswell, W. (1994). "Flocking is an Effective Anti-Predation Strategy in Redshanks (*Tringa Tetanus*)". Animal Behaviour, 47, pp. 433-442.

35. Das, A., Fierro, A., Kumar, V., Ostrowski, J., Spletzer, J. and Taylor, C. (2002). "A Vision-Based Formation Control Framework". IEEE Transactions on Robotics and Automation, 18(5), pp. 813-825.

36. Davis, L. (1991). "Handbook of Genetic Algorithms". VNR Computer Library, New York. (ISBN 0-442-00173-8).

37. De Jong, K.A. (1975). "An Analysis of the Behavior of a Class of Genetic Adaptive Systems". Ph.D. Thesis, University of Michigan, Ann Arbor, USA.

38. De La Cruz, C. and Carelli, I. (2006). "Dynamic Modeling and Centralized Formation Control of Mobile Robots". In Proceeding of 32nd Annual Conference on IEEE Industrial Electronics (IECON 2006, Paris, France), pp. 3880-3885.

39. Deneubourg, J.-L., Pasteels, J.M., and Verhaeghe, J. C. (1983). "Probabilistic Behaviour in Ants: A Strategy of Errors?". Journal of Theoretical Biology, 105, pp. 259-271.

40. Desai, J.P. (2002). "A Graph Theoretic Approach for Modeling Mobile Robot Team Formations". Journal of Robotic Systems, 19(11), pp. 511-525.

41. Dorigo, M., Trianni, V., Sahin, E., Groß, R., Labella, T.H., Baldassarre, G., Nolfi, S., Deneubourg, J.-L., Mondada, F., Floreano, D. and Gambardella, F. (2004). "Evolving Self-Organizing Behaviors for a Swarm-bot". Autonomous Robots, 17(2-3), pp. 223-245.

42. Dorigo, M., Tuci, E., Groß, R., Trianni, V., Labella, T.H., Nouyan, S., Deneubourg, J.-L., Baldassarre, G., Nolfi, S., Mondada, F., Floreano, D., Gambardella, L.M. *et al.*

(2004). "The SWARM-BOTS Project". International Workshop on Swarm Robotics (SAB 2004, Santa Monica, USA, revised selected paper), Lecture Notes in Computer Science 3342, pp. 31-44.

43. Dorigo, M. and Schnepf, U. (1993). "Genetic-Based Machine Learning and Behaviour-Based Robotics: A New Synthesis". IEEE Transactions on Systems, Man and Cybernetics, 23(1), pp. 141-154.

44. Edelstein-Keshet, L. (2001). "Mathematical Models of Swarming and Social Aggregation". In Proceedings of the 2001 International Symposium on Nonlinear Theory and its Applications (NOLTA 2001, Miyago, Japan), pp. 159-164.

45. Franks N.R., Gomez N., Goss S. & Deneubourg J.L. (1991). "The Blind Leading the Blind in Army Ant Raid Patterns: Testing a Model of Self-Organization", Journal of Insect Behavior, 4, pp. 583-607.

46. Esposito, J.M. and Dunbar, T.W. (2006). "Maintaining Wireless Connectivity Constraints for Swarms in the Presence of Obstacles". In Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2006, Florida, USA), pp. 946- 951.

47. Fierro, R. and Das, A.K (2002). "A Modular Architecture for Formation Control". In Proceedings of IEEE the Third International Workshop on Robot Motion and Control (RoMoCo 2002, Poland), pp. 285-290.

48. Fogel, D.B. (1999). "Evolutionary Computation: Toward a New Philosophy of Machine Intelligence". Wiley-IEEE Press, Second Edition (ISBN: 078035379X).

49. Folino, G. and Spezzano, G. (2002). "An Adaptive Flocking Algorithm for Spatial Clustering". In Proceedings of Parallel Problem Solving from Nature (PPSN VII, Granada, Spain), Lecture Notes in Computer Science, 2439, pp.924-933.

50. Folino, G., Forestiero, A. and Spezzano, G. (2003). "Swarming Agents for Discovering Clusters in Spatial Data". In Proceedings of Second International Symposium on Parallel and Distributed Computing (ISPDC 2003, Ljubljana, Slovenia), pp 72-79.

51. Fredlunds, J. and Mataric, M.J. (2002). "A General Algorithm for Robot Formations using Local Sensing and Minimal Communication." IEEE Transactions on Robotics and Automation, 18(5), pp 837-846.

52. Freeman, R.A., Yang, P. and Lynch, K.M. (2006). "Distributed Estimation and Control of Swarm Formation Statistics". In Proceedings of the IEEE 2006 American Control Conference (Minneapolis, USA), pp. 749-755.

53. Fukada, T., Nakaggawa, S., Kawauchi, Y. and Buss, M. (1989). "Structure Decision Method for Self-Organizing Robots Based on Cell Structure – CEBOT". In Proceedings of the 1989 IEEE International Conference on Robotics and Automation (Los Alamitos, USA), pp. 695-700.

54. Garnier, S., Jost, C., Jeanson, R., Gautrais, J., Asadpour, M., Caprari, G. and Theraulaz, G. (2005). "Collective Decision-Making by a Group of Cockroach-Like Robots". In Proceedings of IEEE Swarm Intelligence Symposium (SIS 2005, California, USA), pp. 233-240.

55. Gazi, V. and Passino, K.M. (2004). "Stability Analysis of Social Foraging Swarms". IEEE Transactions on Systems, Man and Cybernetics-Part B Cybernetics, 34(1), pp. 539-557.

56. Gershenson, C. (2002). "Behaviour-Based Knowledge Systems: An Epigenetic Path from Behaviour to Knowledge". In Proceedings of the Second Workshop on Epigenetic Robotics (Edinburgh, UK).

57. Goldberg, D.E. (1989). "Genetic Algorithms in Search Optimization and Machine Learning". Addison-Wesley, Reading, MA, USA, (ISBN: 0201157675).

58. Goldberg, D.E. And Deb, K. (1991). "A Comparative Analysis of Selection Schemes used in Genetic Algorithms". In G. Rawlins (Ed.), Foundations of Genetic Algorithms. Morgan Kaufmann, pp. 69-93.

59. Gordon-Spears, D.F. And Spears, W.M. (2002). "Analysis of a Phase Transition in a Physics-Based Multiagent System". Lecture Notes in Computer Science 2699, pp. 193-207.

60. Gould L. L. and Heppner F. (1974). "The Vee Formation of Canada geese". AUK 91(3), pp. 494-506.

61. Grunbaum, D. and Okubo, A. (1999). "Modelling Social Animal Aggregations". In Levin, S.A. (Ed.), Frontiers in Theoretical Biology, Lecture Notes in Biomathematics, 100, Springer-Verlag, Berlin, pp. 296–325.

62. Gustavi, T. and Hu, X. (2005). "Formation Control for Mobile Robots with Limited Sensor Information". In Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005, Barcelona, Spain), pp. 1791-1796.

63. Hamilton, W.D. (1971). "Geometry for Selfish Herd". Journal of Theoretical Biology, 3, pp. 295-311.

64. Hanada, Y., Lee G. and Chong, N.Y. (2007). "Adaptive Flocking of a Swarm of Robots Based on Local Interactions". In Proceedings of IEEE Swarm Intelligence Symposium (SIS 2007, Hawaii) pp. 340-347.

65. Holland, J.H. (1975). "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence". University of Michigan Press (2nd ed.: MIT Press ,1992), Ann Harbor (ISBN: 0262581116).

66. Holland, O. and Melhuish, C. (1999). "Stigmergy, Self Organization, and Sorting in Collective Robotics", Artificial Life 5(2), pp 173-202.

67. Hornby G.S. and Pollack J.B. (2001). "Evolving L-systems to Generate Virtual Creatures". Computers and Graphics, Elsevier, 25(6), pp. 1041-1048.

68. Howard, H.E. (1929). "An Introduction to the Study of Bird Behaviour". The Geographical Journal 74(5), pp. 504. Cambridge University Press.

69. Ikawa, T. and Okabe, H. (1997). "Three-Dimensional Measurements of Swarming Mosquitoes: a Probabilistic Model, Measuring Systems, and Example Results". In *Animal Groups in Three Dimensions* (Ed. Parrish, J.K. And Harmer, W.M.), Cambridge University Press, pp. 90-104.

70. Javaid, K., Hong, B.R. and Gao, Q.J. (2004). "Dynamic Growth of Robot Formation Using Only Local Sensing and Minimal Communication". In Proceedings of IEEE Eigth International Conference on Control, Automation, Robotics and Vision (ICARCV 2004, Kunming, China) , pp. 283-288.

71. Jones, C. and Mataric, M. (2003). "Adaptive Division of Lab or in Large Scale Minimalist Multi-Robot Systems". In Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS 2003, Las Vegas, USA), pp. 1969-1974.

72. Kaminka, G.A. and Glick R., (2006). "Towards Robust Multi-Robot Formations". In Proceedings of IEEE International Conference on Robotics and Automation (ICRA

2006, Florida, USA), pp. 582-588.

73. Kellogg, J., Bovais, C., Foch, R., McFarlane, H., Sullivan, C. and Dahlburg, J. (2002). "The NRL Micro Tactical Expandable (MITE) Vehicle". The Aeronautical Journal, 106.

74. Kennedy, J. and Eberhart, R.C. (1995). "Particle Swarm Optimization". In Proceedings of IEEE International Conference on Neural Networks (Piscataway, NJ. USA), pp. 1942-1948.

75. Kennedy, J., Eberhart R.C. and Shi, Y. (2001). "Swarm Intelligence". The Morgan Kaufmann Series in Artificial Intelligence (ISBN: 1558605959).

76. Klein, J. (2002). "Breve: A 3D Environment for the Simulation of Decentralized Systems and Art". In Proceeding of Artificial Life VIII, the Eight International Conference on the Simulation and Synthesis of Living Systems, The MIT Press, pp. 329-334.

77. Kodati, P., Hinkle, J. and Deng, X. (2007). "Micro Autonomous Robotic Ostraciiform (MARCO): Design and Fabrication". In Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2007, Italy), pp. 960-965.

78. Kókai, G., Ványi, R. and Toth, Z. (1999). "Parametric L-System Description of the Retina with Combined Evolutionary Operators". In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999, Florida, USA), 2, pp. 1588-1595.

79. Krause, J. (1994). "Differential Fitness Returns in Relation to Spatial Position in Groups". Biological Reviews, Cambridge Philosophical Society, 69(2), pp.187-206.

80. Kube, C. and Zhang, H. (1992). "Collective Robotic Intelligence". In Proceedings of Second International Conference on Simulation of Adaptive Behaviour (SAB 1992, Hawaii, USA), MIT Press, pp.460-468.

81. Langton, C. (1988). "Artificial Life". In Artificial Life, 6. Sante Fe Institute Studies in the Sciences of Complexity, Addison Wesley.

82. Lee, D. and Yannakakis, M. (1996). "Principles and Methods of Testing Finite State Machines - A Survey". In the Proceeding of IEEE, 84(8), pp. 1089-1123.

83. Levine, H., Rappel, W-J. And Cohen, I. (2000). "Self-Organization in Systems of Self-Propelled Particles". Physical Review E, 6, p. 017101.

84. Li, Y, Yuan, K. and Zou, W. (2006). "Nonholonomic Mobile Robot Formation Control

with Kinodynamic Constraints". In Proceedings of the 2006 International Symposium on Practical Cognitive Agents and Robots (PCAR 2006, Perth, Australia), pp. 200-211.

85. Lindenmayer, A. (1968). "Mathematical Models for Cellular Interaction in Development, Part I and Part II". Journal of Theoretical Biology, 18, pp. 280-315.

86. Lissaman P.B., Shollenberger C.A. (1970). "Formation Flight of Birds". Science 168(3934), pp. 1003-1005.

87. Liu Y, Passino K.M. (2000). "Swarm Intelligence: Literature Overview". Department of Electrical Engineering, The Ohio State University, March 2000, http://www.ece.osu. Edu/~passino/swarms.pdf

88. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G. (2005). "MASON: A Multiagent Simulation Environtment". Simulation 81(7), pp. 517-527.

89. Maes, P. (1995). "Modelling Adaptive Autonomous Agents". Artificial Life 1(1-2), The MIT Press, pp. 135-162.

90. Mariano, A., Moscato, P. and Norman, M.G. (1995). "Using L-Systems to Generate Arbitrarily Large Instances of the Euclidean Traveling Salesman Problem with Known Optimal Tours". Anales del XXVII Simposio Brasileiro de Pesquisa Operacional, Vitoria, Brasil.

91. Mastellone, S., Stipanovic, D.M. and Spong, M.W. (2007). "Remote Formation Control and Collision Avoidance for Multi-Agent Nonholonomic Systems". In Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2007, Italy), pp. 1062-1067.

92. Mataric, M.J. (1999). "Behavior-Based Robotics". MIT Encyclopedia of Cognitive Sciences, Robert A. Wilson and Frank C. Keil, eds., MIT Press, pp. 74-77.

93. Mataric, M.J. (1993). "Designing Emergent Behaviors: From Local Interactions to Collective Intelligence". In Proceedings, From Animal to Animats, Second International Conference on Simulation of Adaptive Behavior (SAB 1992, Hawaii), MIT Press, pp 432-441.

94. MASON (2007). "MASON: Multiagent Simulation Environment". [online]. Last accessed on 18 April 2007 at http://www.cs.gmu.edu/~eclab/projects/mason/

95. Mealy, G. H. (1955). "A Method to Synthesizing Sequential Circuits". Bell System

Technical J, pp. 1045-1079.

96. Michel, O. (2004). "Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation". International Journal of Advanced Robotic Systems 1(1), pp. 39-42.

97. Mitchell, M. (1996). "An Introduction to Genetic Algorithms". MIT Press, Cambridge, MA, USA (ISBN: 0262631857).

98. Mogilner, A. and Edelstein-Keshet, L (1999). "A Non-Local Model for a Swarm". Journal of Mathematical Biology 38(6), pp. 534-570.

99. Mogilner, A., Edelstein-Keshet, L., Bent, L. and Spiros, A. (2003). "Mutual Interactions, Potentials, and Individual Distance in a Social Aggregation". Journal of Mathematical Biology, 47(4), pp. 353-389.

100. Momen, S., Amavasai, B.P. and Siddique, N.H. (2007). "Mixed Species Flocking for Heterogeneous Robotic Swarms". In Proceeding of The International Conference on Computer as a Tool (EUROCON 2007, Poland), pp: 2329-2336.

101. Moore, E.F. (1956). "Gedanken-Experiments on Sequential Machines". Automata Studies, Annals of Mathematical Studies, 34, Princeton University Press, Princeton, N.J., pp. 129–153.

102. Moshtagh, N., Jadbabaie, A. and Daniilidis, K. (2006). "Vision-Based Control Laws for Distributed Flocking of Nonholonomic Agents". In Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2006, Florida, USA), pp. 2769- 2774.

103. NetLogo (2007). "NetLogo: Multi-Agent Programmable Modelling Environment". [online]. Last accessed on 15 April 2007 at http://ccl.northwestern.edu/netlogo/

104. Newmann, V. (1966). "Theory of Self-Reproducing Automata". (Edited & Completed by Burks, A.W.) University of Illinois Press.

105. Nguyen, A.D., Ha, Q.P., Huang, S. and Trinh, H. (2004). "Observer-Based Decentralized Approach to Robotic Formation Control". In Proceedings of Australasian Conference on Robotics and Automation (ACRA 2004, Canberra, Australia).

106. Nguyen, D.B. and Do, K.D. (2006). "Formation Control of Mobile Robots". International Journal of Computers, Communications & Control, 1(3), pp.41-59.

107. Nouyan, S., Groß, R., Bonani, M., Mondada, F., Dorigo, M. (2006). "Group Transport

Along a Robot Chain in a Self-Organised Robot Colony". In Proceeding of the Ninth International Conference on Intelligent Autonomous Systems, IOS Press, Amsterdam, The Netherlands, pp. 433-442.

108. Olfati-Saber, R., Murray, R.M. (2003). "Flocking with Obstacle Avoidance: Cooperation with Limited Communication in Mobile Networks". In Proceedings of the 42nd IEEE Conference on Decision and Control (Hawaii, USA), 2, pp.2022-2028.

109. Othman, W.F.W., McKibbin, S.P., Caparrelli, F., Travis, J.R. and Amavasai, B.P. (2005). "Pattern Formation and Organisation in Robot Swarms". In Proceedings of the IEEE SMC UK-RI Chapter Conference on Applied Cybernetics (London, UK), pp. 135-140,

110. Parker, L.E., Kannan, B., Tang, F. and Bailey, M. (2004). "Tightly-Coupled Navigation Assistance in Heterogeneous Multi-Robot Team". In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004, Sendai, Japan), pp. 1016-1022.

111. Parrish, J.K. and Edelstein-Keshet, L. (1999). "Complexity, Pattern, and Evolutionary Trade-Offs in Animal Aggregation". Science, 284, pp. 99-101.

112. Parrish, J.K., Viscido, S.V. And Grunbaum, D. (2002). "Self-Organized Fish Schools: An Examination of Emergent Properties". Biological Bulletin, 202, pp. 296-305.

113. Pavone, M. and Frazzoli, E. (2007). "Decentralized Policies for Geometric Pattern Formation". In Proceedings of the 2007 American Control Conference (ACC 2007, New York, USA), pp.3949-3954.

114. Payton, D., Estkowski, R. and Howard, M. (2004). "Pheromone Robotics and the Logic of Virtual Pheromones". International Workshop on Swarm Robotics (SAB 2004, Santa Monica, USA, revised selected paper), Lecture Notes in Computer Science 3342, pp. 45-57.

115. Penders, J., Alboul, L., Roast, C. and Cervera, E. (2007). "Robot Swarming in the Guardians Project". In Proceeding of European Conference on Complex Systems 2007 (ECCS 2007, Dresden, Germany).

116. Powell, G.V.N. (1974). "Experimental Analysis of the Social Value of Flocking by Starlings (*Sturnus vulgaris*) in Relation to Predation and Foraging". Animal Behaviour, 22, pp. 501-505.

117. Prusinkiewicz, P. and Lindenmayer, A. (1990). "The Algorithmic Beauty of Plants". Springer-Verlag.

118. Rabiner, L., Juang, B. (1996). "An introduction to Hidden Markov Models". ASSP Magazine, IEEE, Jan 1986, 3(1), pp. 4-16.

119. Railsback, S. F., Lytinen, S.L. and Jackson, S.K. (2006). "Agent-Based Simulation Platforms: Review and Development Recommendations". Simulation 82(9), pp. 609-623.

120. Reynolds, C. (1987). "Flocks, Herds and Schools: A Distributed Behavioral Model". ACM SIGGRAPH Computer Graphics 6(4), pp. 25-34.

121. Rune, V., Nøttestad, L. (1997). "An Individual Based Model of Fish School Reactions: Predicting Antipredator". Fisheries Oceanography 6(3) pp. 155-171.

122. Russell, S. and Norvig, P. (1995). "Artificial Intelligence: A Modern Approach". Pearson Education Inc., ISBN (0-131-03805-2).

123. Sahin, E. (2005). "Swarm Robotics: From Sources of Inspiration to Domains of Application". In Proceedings of International Workshop on Swarm Robotics (SAB 2004, Santa Monica, USA, revised selected paper), Lecture Notes in Computer Science, Swarm Robotics, 3342, pp. 10-20.

124. Salvador, P., Nogueira, A. and Valadas, R. (2002). "Modeling Multifractal Traffic with Stochastic L- Systems". In: Proceedings of IEEE Global Telecommunication Conference 2002 (GLOBECOM 2002, Taipei, Taiwan), 3, pp. 2518-2522.

125. Schmickl, T. and Crailsheim, K. (2007a). "A Navigation Algorithm for Swarm Robotics Inspired by Slime Mold Aggregation". Second International Workshop on Swarm Robotics (SAB 2006, Rome, Italy), Lecture Notes in Computer Science 4433, pp. 1-13.

126. Schmickl, T. and Crailsheim, K. (2007b). "Trophallaxis within a Robotic Swarm: Bio-Inspired Communication among Robots in a Swarm". Autonomous Robot, Springer Netherlands, 25(1-2), pp. 171-188.

127. Shao, J., Wang, L. and Yu, J. (2006). "Underwater Transportation of Multiple Fish-like Robots using Situation Based Action Selection". In Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2006, , Florida, USA), pp. 3208- 3213.

128. Shaw, E. (1962). "The Schooling of Fishes". Scientific American, 206, pp. 128-138.

129. Shimoyama, N., Sugawara, K., Mizuguchi, T., Hayakawa, Y. and Sano, M. (1996). "Collective Motion in a System of Motile Elements". Physical Review Letters 76(20), pp. 3870-3873.

130. Smith, A. (1984). "Plants, Fractals, and Formal Languages". ACM SIGGRAPH Computer Graphics, 18(4), pp. 1-10.

131. Sorensen, N. and Ren, W. (2007). "A Unified Formation Control Scheme with a Single or Multiple Leaders". In Proceedings of the 2007 American Control Conference (ACC 2007, New York, USA), pp. 5412-5418.

132. Spears, W. and Gordon, D. (1999). "Using Artificial Physics to Control Agents". In Proceedings of IEEE International Conference on Information, Intelligence, and Systems (Maryland, USA), pp. 281-288.

133. Spears, W.M., Spears, D.F., Heil, R., Kerr, W. and Hettiarachchi, S. (2005). "An Overview of Physicomimetics". International Workshop on Swarm Robotics (SAB 2004, Santa Monica, USA, revised selected paper), Lecture Notes in Computer Science 3342, pp. 84-97.

134. Spears, D., Zarzhitsky, D. and Thayer, D. (2005). "Multi-Robot Chemical Plume Tracing". In proceedings of the 2005 International Workshop on Multi-Robot Systems, From Swarms to Intelligent Automata Volume III (Washington DC, USA), Springer Netherlands, pp. 211-221.

135. Spector, L., Klein, J. and Keijzer, M. (2005a). "The Push3 Execution Stack and the Evolution of Control". In proceeding of the Genetic and Evolutionary Computation Conference (GECCO 2005, Washington DC, USA), Springer-Verlag, pp. 1689-1696.

136. Spector, L., Klein, J., Perry C. and Feinstein, M. (2005b). "Emergence of Collective Behavior in Evolving Populations of Flying Agents". Genetic Programming and Evolvable Machine 6(1), pp. 111-125.

137. Steels, L. (1991). "Towards a Theory of Emergent Functionality". In Proceedings of the First International Conference on Simulation of Adaptive Behaviour: From Animal to Animats (SAB 1991, Paris, France), MIT Press, pp. 451-461.

138. Sun, D. and Wang, C. (2007). "Controlling Swarms of Mobile Robots for Switching between Formations Using Synchronization Concept". In Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2007, Italy), pp. 2300-

2305.

139. Takahashi, H., Nishi, H. and Onishi, K. (2004). "Autonomous Decentralized Control for Formation of Multiple Mobile-Robots Considering Ability of Robot". IEEE Transactions on Industrial Electronics 51(6), pp. 1272-1279.

140. Tanner, H. G., Jadbabaie, A. and Pappas G. J. (2003a). "Stable Flocking of Mobile Agents, Part I: Fixed Topology". In Proceedings of the 42nd IEEE Conference on Decision and Control (CDC 2003, Maui, Hawaii), pp. 2010-2015.

141. Tanner, H. G., Jadbabaie, A. and Pappas G. J. (2003b). "Stable Flocking of Mobile Agents, Part II: Dynamic Topology". Proceedings of the 42nd IEEE Conference on Decision and Control (CDC 2003, Maui, Hawaii), pp. 2016-2021.

142. Tanner, H.G. and Kumar, A. (2005). "Towards Decentralization of Multi-robot Navigation Functions". In Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005, Barcelona, Spain), pp. 4132-4137.

143. Taylor, C.E. (1990). "Fleshing Out". Artificial Life II: Proceedings of the Workshop on Artificial Life II, Addison-Wesley Press, pp. 25-38.

144. Terzopoulos, D., Tu, X. and Grzeszczuk, R. (1994). "Artificial Fishes with Autonomous Locomotion, Perception, Behaviour, and Learning in a Simulated Physical World". Artificial Life, 1(4), pp. 327-351.

145. Tisue, S. and Wilensky, U. (2004). "NetLogo: A Simple Environment for Modelling Complexity". In Proceeding of International Conference on Complex Systems (ICCS 2004, Boston, MA, USA).

146. Toner, J. and Tu, Y. (1998). "Flocks, Herds, and Schools: A Quantitative Theory of Flocking". Physical Review E, 58(4), pp. 4859-4864.

147. Turing, A.M. (1950). "Computing Machinery and Intelligence". Mind: A Quarterly Review of Psychology and Philosophy, 59(236), pp.433-460.

148. Vabø, R. and Nøttestad, L. (1997). "An Individual Based Model of Fish School Reactions: Predicting Antipredator". Fisheries Oceanography, 6(3), pp. 155-171.

149. Veeraswamy, A., Amavasai, B.P. And Meikle, S. (2006). "Optimal Path Planning Applied to Ant Foraging". In Proceedings of the IEEE Systems, Man and Cybernetics UK-RI Chapter Conference on Advances in Cybernetic Systems, pp. 300-305.

150. Viscek, T., Czirok, A., Ben-Jacob, E., Cohen, I. and Shochet, O. (1995). "Novel Type of Phase Transition in a System of Self-Driven Particles". Physical Review Letters 75(6), pp. 1226-1229.

151. Von-Frisch, K. (1968). "The Dance Language and Orientation of Bees". Harvard University Press, (ISBN: 0674190513).

152. Walter, W.G. (1950). "An Imitation of Life". Scientific American, 182(5), May 1950, pp. 42-45.

153. Walter, W.G. (1963). "The Living Brain". W. W. Norton & Company, Inc. (ISBN: 0393001539)

154. Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

155. Wilensky, U. (1998). "NetLogo Flocking Model". http://ccl.northwestern.edu/netlogo/models/Flocking, Center for Connected Learning and Computer Based Modelling, Northwestern University, Evanston IL, 1998.

156. Woern, H., Szymanski, M. and Seyfried, J. (2006). "The I-SWARM Project". In Proceedings of The 15th IEEE International Symposium on Robot and Human Interactive Communication, (ROMAN 2006, United Kingdom), pp. 493-496.

157. Yang, T., Liu, Z., Chen, H. and Pei, R. (2007). "Robust Tracking Control of Mobile Robot Formation with Obstacle Avoidance". Journal of Control Science and Engineering 2007 (ID 51841), Hindawi Publishing Corporation, DOI:10.1155/2007/51841.

158. Zamir, M (2001). "Arterial Branching within the Confines of Fractal L-System Formalism". The Journal of General Physiology, 118(3), pp. 267-276.

159. Zarzhitsky, D., Spears, D.F. and Spears, W.M. (2005). "Distributed Robotics Approach to Chemical Plume Tracing". In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005, Canada), pp. 4034- 4039.

160. Zhang, Z.G., Gondo, M., Yamashita, N., Yamamoto, A. and Higuchi, T. (2007). "Design and Control of a Fish-like Robot Using an Electrostatic Motor". In Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2007, Italy), pp. 974-979.

Publications

1. McKibbin, S.P., Amavasai, B., Selvan, A.N., Caparrelli, F., Othman, W.A.F.W. (2008). "The Role Of Sensory-Motor Coordination: Identifying Environmental Motion Dynamics with Dynamic Neural Networks". In Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2008, Madeira, Portugal).

2. McKibbin, S.P., Amavasai, B., Selvan, A.N., Caparrelli, F., Othman, W.A.F.W. (2007) "Recurrent Neural Robot Controllers: Feedback Mechanisms for Identifying Environmental Motion Dynamics". AIRE, 2-3(27), pp. 113-130.

3. McKibbin, S.P., Amavasai, B.P., Caparrelli, F., Othman, W.A.F.W. and Travis, J.R. (2007). "Using PSO to Exploit Movement as a Sensory Cue in Autonomous Mobile Robots". In Proceedings of the 2007 IEEE SMC UK-RI 6th Chapter Conference on Cybernetic Systems (CS 2007, Dublin, Ireland), pp. 152-157.

4. Othman, W.A.F.W., Amavasai, B.P., McKibbin, S.P. & Caparrelli, F. (2007). "An Analysis of Collective Movement Models for Robotic Swarms". In Proceeding of The International Conference on Computer as a Tool (EUROCON 2007, Poland), pp. 2373-2380.

5. McKibbin, S.P., Caparrelli, F., Othman, W.F.W., Amavasai, B.P, & Travis, J.R. (2006). "Complexity in Evolving Neural Robot Controllers". In Proceedings of the 2006 IEEE SMC UK-RI 5th Chapter Conference on Advances in Cybernetic Systems (AICS 2006, Sheffield, UK).

6. Othman, W.F.W., Amavasai, B.P., McKibbin, S.P., Caparrelli, F. & Travis, J.R. (2006). "Evolutionary L-Systems for Large Scale Multi-Robot Formation". In Proceedings of the 2006 IEEE SMC UK-RI 5th Chapter Conference on Advances in Cybernetic Systems (AICS 2006, Sheffield, UK), pp. 199-204.

7. Fernandez, J.M., Amavasai, B.P., Othman, W.F.W., McKibbin, S.P., Caparrelli, F. &

Travis, J.R. (2005). "Development of Physical Agents for Robot Swarms". In Proceedings of the 2005 IEEE SMC UK-RI Chapter Conference on Applied Cybernetics (London, UK), pp. 123-128.

8. Othman, W.F.W., McKibbin, S.P., Caparrelli, F., Travis, J.R. & Amavasai, B.P. (2005). "Pattern Formation and Organisation in Robot Swarms". In Proceedings of the 2005 IEEE SMC UK-RI Chapter Conference on Applied Cybernetics (London, UK), pp. 135-140.

9. McKibbin, S.P., Othman, W.F.W., Amavasai, B.P., Caparrelli, F. & Travis, J.R. (2004). "Designing Microrobot Swarms - Issues and Constraints". In Proceedings of the 2004 IEEE SMC UK-RI 3rd Chapter Conference on Intelligent Cybernetic Systems (ICS 2004, Londonderry, UK), pp. 80-84.