Study of an Adaptive and Multifunctional Computational Behaviour Generation Model for Virtual Creatures

Fang Wang

Doctor of Philosophy

Institute of Perception, Action and Behaviour
Division of Informatics
University of Edinburgh

2002

Abstract

High fidelity virtual environments can be inhabited by virtual living creatures. A virtual creature should be able to learn by itself how to improve its old behaviours and produce new related behaviours so as to be more adaptive and autonomous and hence reduce human design work. This thesis presents a study of an adaptive and multifunctional Computational Behaviour Generation (CBG) model for virtual creatures with the ultimate goal of enhancing a creature's adaptation and multifunctionality in behaviour control by learning. Specifically, we require that the CBG model can learn to perform variable behaviours in various environments and situations.

The design of the CBG model is inspired by the natural behaviour control system in the brain. It can perform the whole procedure of selection, programming and execution of motor actions, and its hierarchical architecture provides the material basis for adaptive and multifunctional learning implementation. The concrete achievement of adaptation and multifunctionality by learning is obtained with the help of a Multi-agent based Evolutionary artificial Neural network with Lifetime learning (MENL), which can learn to select suitable motor actions for varied behaviours in varied situations. MENL maintains a batch of agents in every evolutionary generation to co-decide the actions to be executed. These agents are subject to evolutionary learning through all of their lifetime. The fitness function of MENL is designed without many specific constraints, and can be easily extended for a variety of behaviours. In consequence the CBG with MENL can obtain high adaptation and generalisation in behaviour.

The CBG model combined with the MENL learning algorithm enables a virtual creature to learn several space occupying behaviours independently and jointly in unknown environments. These behaviours are exploration, goal reaching, and wandering. The virtual creature is first asked to learn exploration only in a series of increasingly complex environments. This creature adapts to various environments and explores them successfully. The successful exploration experiment is achieved due to the competition and cooperation among multiagents and their continuous lifetime learning. Inspired by multifunctional neural networks in the natural behaviour control, the CBG combined with MENL is then required to learn exploration and goal reaching jointly, and exploration, goal reaching, and wandering jointly in some unknown environments. Experimental results have shown that the CBG with MENL can perform multiple space occupying behaviours competently. Moreover, the overall performance of the multifunctional learning is better than that of the sum of learning every behaviour independently.

The research presented in this thesis leads to the conclusion that the CBG model and the MENL learning algorithm, which are inspired by the biological neural mechanisms for behaviour, are useful methods for achieving adaptive and multifunctional behaviour control for virtual creatures.

Acknowledgements

Many people contributed in some way or other to the work presented in this thesis. The first acknowledgement must go to my supervisor Eric Mckenzie for his always kind support during my study. I am indebted to my second supervisor Gillian Hayes for her valuable suggestions and comments on the thesis. Without her help, the thesis would not be as it is. I also would like to thank John Hallam, Jean-Arcady Meyer and Robert Fisher for their insightful comments on my work.

I am grateful to Roland Ibbett and John Power who have helped me in many ways on both studying and living in the UK. Thanks to Heather Maclaren, Alex Champandard, Jonathan Meddes, Rangsipan Marukatat, and Richard Reeve, for many enjoyable discussions and useful comments. Thanks to my other friends for their support and great fun.

My most special thanks are to my parents and my husband for their love, understanding and encouragement throughout this time.

This research was supported by Falconer Memorial Faculty of Science and Engineering Scholarship at the University of Edinburgh and a studentship from the British Overseas Research Scheme. Some revision work of this thesis was conducted at BTexact Technologies.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Some parts of this work are based on previously published papers [166, 167, 168, 169].

(*Fang Wang*)

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Virtual environments are at the heart of virtual reality. They are computer-generated worlds that the human user can participate in and interact with. For enhanced fidelity virtual environments can be designed to be inhabited by virtual living creatures. For constructing realistic virtual lives in virtual environments, the generation of believable behaviours for these lives is a very important factor. This is because the human user recognises not only form, colour and sound, but also changes in behaviours [9]. Additionally, manipulation of a virtual life is only realistic if this life responds appropriately to the user's actions. Although believable behaviours are so important to virtual lives, how to attach them to computer-generated virtual lives in real-time is a thorny problem that has troubled researchers for a long time.

Traditional approaches to creating moving virtual objects by computers are to employ skilled artists and animators to generate an animation off-line and then present it with existing graphics techniques. By using animation techniques such as keyframe, kinematics, and dynamics, skilled human animators have produced very realistic characters in Disney animation and many movies like *Batman* and *Terminator*. However, the very intensive and time-consuming labour of human animators in specifying every motion of the characters makes traditional

animation unsuitable to real-time virtual environments. Most importantly, the animated characters created in traditional animation are very rigid and fixed objects which have only the move sequences predefined by human animators. The behaviours of these characters can only be created beforehand and replayed afterwards. They are not flexible, re-organisable, and reusable behaviours that are able to adjust to various changes spontaneously. It is obvious that those completely hand-crafted animated characters are not true virtual lives that are capable of living in continuously changing virtual environments.

Recent development in *Behavioural Animation* [133] is a new attempt to give virtual lives real-time, believable behaviours. The key idea of behavioural animation is to impart to virtual lives useful knowledge about behaviours so that the lives can decide themselves when, where, and how to move their bodies appropriately according to environmental conditions. This is just what real lives do in nature. Therefore, virtual lives in behavioural animation are actually self-animated by simulating some of the natural mechanisms fundamental to life. To date, a series of virtual lives have been produced in both academic and commercial areas, including virtual plants, virtual animals, and even virtual humans. Some representative examples are the "boids" of Reynolds [133], artificial fish of Terzopoulos et al. [162], and the synthetic dog of Blumberg [20]. These virtual lives have presented some "life" characteristics by simulating the real lives from different directions. Some virtual lives, for example, can grow up and this growth is somewhat affected by certain environmental changes. Some virtual creatures can move in a virtual environment based on their perceived environmental information. In addition, they can accomplish some simple tasks when moving around. Some creatures have enhanced believability and even friendliness when emotional state and changes are produced. Compared with traditional animated characters, the virtual lives created by behavioural animation have exhibited much more flexibility and autonomy.

Nevertheless, the design of previous virtual lives suffers from several serious limitations. One important limitation is that these virtual lives lack an efficient learning capability to improve their behaviours continuously according to knowl-

edge gleaned by themselves. Although some virtual lives are endowed with certain learning skills, the behaviours of the previous virtual lives heavily depend on human design and are not improvable once they are set or learned. It is usually the human designer that tells the virtual lives what to do in what situations by providing some behaviour control rules. The virtual lives can execute or learn a few behaviours in some recommended conditions, but they may not be able to adjust to new, complex, or even unknown environmental changes by using their own knowledge. In consequence the behaviours of these virtual lives are in fact confined to a scope prescribed by the human designer. Because virtual environments are real-time environments that are dynamically changed by the participation of the human user, and because these changes are not always predictable, it is questionable whether these virtual lives can survive the changes in virtual environments. In particular, the virtual lives may face a "fatal" problem in some urgent, unexpected situations, if they cannot learn to improve their behaviours so as to find a way out of their difficulties by themselves.

Another limitation in previous work on virtual lives lies in their incapacity to reuse knowledge and mechanisms for generating more than one behaviour. Generally, those virtual lives require the human designer to carefully design necessary executing details of every designated behaviour, even when the common knowledge and shared mechanisms across some related behaviours have been given to the lives already. It is obvious that, if the virtual lives can reuse the shared mechanisms and transfer the knowledge they have learned from one behaviour for the implementation of other behaviours, the generation of new behaviours would be much easier and more efficient in both space and time. This kind of multifunctionality of performing several behaviours together is common to real lives that can produce numerous behaviours in a limited anatomical structure, by sharing many neural circuits in executing functionally related behaviours [43, 122]. Due to the lack of knowledge sharing and transference, the design effort in designing behaviours for virtual lives is usually very high. Moreover, the adaptive and autonomous ability of the virtual lives is limited too. Ideally, a virtual life should be able to learn by itself how to apply what it acquires from one behaviour to an-

other if there is something in common in these behaviours. For achieving powerful adaptation and autonomy in virtual life behaviours, there is still much remaining to be done.

1.2 Aims

This thesis presents an adaptive and multifunctional Computational Behaviour Generation model for virtual lives, and in particular, for virtual creatures, in an attempt to address the above limitations in recent work on virtual creatures. (We are especially interested in the behaviours of virtual creatures because creatures are always much more active and flexible relative to plants, and their behaviours may demonstrate our behaviour model more easily and clearly.) By being adaptive, we hope that a virtual creature is able to learn to cope with new, complicated, unpredictable situations encountered during its lifetime, by using the information available in its environment without human intervention. With the multifunctionality property, a virtual creature should learn to perform multiple functionally related behaviours with efficiency and economy in time and space, by taking advantage of the shared knowledge and mechanisms across these behaviours. With efficient adaptive and multifunctional learning ability, virtual creatures can have enhanced adaptation and autonomy in their behaviours.

Given our objective to construct adaptive and multifunctional behaviours in virtual creatures, is it possible to develop a behaviour generation model that learns to improve its behaviours continuously, and learns multiple behaviours more efficiently than sequential learning? Our answer to this question is definitely positive. We propose a hierarchical Computational Behaviour Generation (CBG) model and a novel learning algorithm, Multi-agent based Evolutionary artificial Neural network with Lifetime learning (MENL), to achieve adaptation and multifunctionality in virtual creatures. As the CBG model provides the necessary mechanisms, the MENL learning algorithm learns correct decision-making of motor action selection for various behaviours in various environments. The generated behaviours are adaptive and multifunctional. The principles underlying

the behaviour model and the learning algorithm will be introduced in Chapter 3 and Chapter 4 respectively. We will demonstrate that this behaviour model combined with the learning algorithm works in adaptive learning in Chapter 4 and in multifunctional learning in Chapter 5.

1.3 Methodology

During the construction of the adaptive and multifunctional Computational Behaviour Generation model, many ideas have been drawn from biology (especially neuroscience) and other fields of computer science, such as agents, artificial neural networks, evolutionary algorithms, robotics, and so on.

The architecture of the Computational Behaviour Generation (CBG) model itself is inspired by recent studies of the natural behaviour control system in the brain [79, 29]. In particular, the hierarchical structure of the natural behaviour control system, the input from body states and environmental information, the adaptation by learning, and the generality in various behaviour systems are the main sources for us to build the CBG model. The CBG model finally constructed holds a mixture of top-down and bottom-up control information flows, which is similar to the natural behaviour control system. In consequence the CBG model is able to utilise the bi-directional information flows to compare its intended actions with actual executed results, and hence learn to improve its behaviours from both successful and unsuccessful experiences. Because the CBG model has a similar hierarchy to the natural behaviour control system and because this hierarchy maintains considerable generality for different behaviour systems, the CBG model also has the capability to produce multiple behaviours in the same structure. The architecture provides the CBG model with the potential strengths of adaptation and multifunctionality.

We propose a Multi-agent based Evolutionary artificial Neural network with Lifetime learning (MENL) to instantiate the latent power of adaptation and multifunctionality in the CBG model. MENL is composed of an artificial neural network evolved by Evolutionary Strategies (ESs). It uses multi-agent system

technology to maintain the diversity of solutions. Meanwhile, MENL adopts a concept of lifetime learning to achieve generalisation in solutions, and employs a relaxed and general fitness function design to search for suitable solutions in a broad space. When the MENL learning algorithm is introduced into the CBG model to select suitable motor actions, the CBG model generates improved behaviours continuously, which are adaptive to varied situations and environments.

The implementation of multifunctionality in the CBG model combined with MENL is also inspired by the natural behaviour control system. In nature, there are many multifunctional neural networks sharing some or even all of their neurones to perform different but functionally related behaviours. Because the fitness function designed in the MENL learning algorithm involves both general sensory feedback and specific behavioural objectives, MENL can easily learn several action selection policies for implementing several related behaviours in the same behaviour control model. These are the hierarchical architecture of the CBG model and the common knowledge shared across related behaviours that help new behaviours to be learned and implemented smoothly based on previously learned behaviours.

The potential of the adaptive and multifunctional Computational Behaviour Generation model is demonstrated by learning several space occupying behaviours[1] that are common and fundamental to natural animals. Using the CBG model combined with the MENL learning algorithm, a virtual creature has learned many space occupying behaviours, including exploration, goal reaching and wandering, both independently and jointly in various unknown environments.

1.4 Organisation of the Thesis

In the following chapters, we will present and demonstrate our proposed adaptive and multifunctional Computational Behaviour Generation (CBG) model in detail. Research work relevant to virtual lives, the CBG model, and the MENL learning

[1]Behaviours like exploration, navigation, wandering, etc., which involve movement from one place to another, are called space occupying behaviours in this thesis. The corresponding movement is space occupying movement. See Section 3.5 for more information.

algorithm will be introduced. Because the design of the CBG model, MENL, and their adaptive and multifunctional learning has touched upon many different areas in neuroscience and computer science, a review of the research work related to these topics will be delivered in specific chapters when their design is explained.

The rest of the thesis is structured as follows. Chapter 2 contains an overview of past work on believable behaviours of virtual lives. It is followed by a brief discussion on our opinions of virtual lives in virtual environments. This includes a working definition of *Virtual Life* and some key issues which we think are important to the construction of virtual lives. This chapter helps us have a basic understanding of virtual lives. It also shows how our work fits into virtual life research as a whole.

Chapter 3 describes the architecture of the Computational Behaviour Generation model. It first summarises the fundamental concepts in biological neural control for behaviours, and then presents the CBG model based on these concepts. Potential strengths of the CBG model including adaptation, multifunctionality, and some others are discussed next. An instantiation of the CBG model for generating space occupying behaviours is introduced too.

Chapter 4 presents the novel learning algorithm of MENL and shows how the CBG model works with MENL to learn single behaviours adaptively. Natural space occupying behaviours, which are fundamental to natural creatures, are used as sample behaviours for the CBG and MENL to learn. In this chapter, a series of experiments on learning exploration in various unknown environments has been conducted for demonstrating the adaptive learning ability of the CBG with MENL. There are several natural and robust behaviours emergent from these experiments.

Empirical experiments on demonstrating the multifunctional learning ability of the CBG and MENL are shown in Chapter 5. Several space occupying behaviours, such as exploration, goal reaching, and wandering, have been learned in an integrated way in the same behaviour control model and the same MENL. Multifunctional learning has shown better learning performance when compared with learning every behaviour independently. Moreover, many interesting nav-

igation trajectories have illustrated the reliability and continuously improving capability of multifunctional learning.

The adaptive and multifunctional CBG model presented in this thesis is finally concluded in Chapter 6, with a reflective look at what has been accomplished and some promising directions for future work.

Chapter 2

Virtual Life in Virtual Environments

In this chapter, we give an introduction to the study of *Virtual Life*. We first point out the importance of presenting virtual lives in virtual environments, and then present a review of previous work on virtual lives. In order to get a deeper understanding of virtual life, we analyse the concept of *Virtual Life* and how to construct believable virtual lives in virtual environments. In particular, we suggest that the behaviours of a virtual life should be at least autonomous, adaptive, and interactive. Following this discussion is a simple comparison between virtual lives and real lives. Next, we outline our work by putting it in the context of virtual life research.

2.1 Presentation of Virtual Life

Virtual Reality (VR) has become a conspicuous technology since its introduction around 1985, due to its ability to provide the effect of immersion in a realistic computer-generated world [31]. Through the use of various sensorial channels (these channels could be visual, auditory, tactile, smell, taste, etc), the human user can sense and interact within this world in real time, in which virtual objects have spatial presence. This computer-generated world providing realistic computer simulation and real-time interaction is called a *Virtual Environment* [30].

Virtual environments play a very important role in virtual reality. They are

the virtual place to immerse the human user and the main sphere for his activity. Convincing immersive virtual reality requires high fidelity virtual environments [9]. To provide high realism in virtual environments, we need to solve a series of key issues, including the ease of interaction, the high quality of the visual, auditory, and other sensorial presentation, and the realistic presence of virtual objects in virtual environments. The virtual objects presented in virtual environments include not only lifelike static objects, such as sky, mountains and buildings, but also vivid and active lives, such as plants, animals and even humans. Those simulated lives, called *virtual lives*, will help virtual environments to be more dynamic and similar to the real world. Like natural living things, virtual lives have lifelike visual shapes and appearances, and believable behaviours. They inhabit the virtual world subject to physical laws. They can receive information from the outside, behave in the virtual world naturally, and interact with real people and other virtual lives properly. They may grow, reproduce, and die. They may even have their own beliefs, desires, and intentions. From virtual plants to virtual creatures, from virtual insects to virtual humans, virtual lives with different complexity will have different characteristics, abilities, and intelligence. A timid virtual animal, for example, runs away when perceiving a human presence, but a virtual pet will run to the person enthusiastically, and the virtual human will talk to it with gestures and emotions. With realistic and dynamic virtual lives, a virtual environment has enhanced fidelity, and helps to maintain the illusion in the user that it is real.

2.2 Previous Work

In this section, we review the work on modelling virtual lives that have life characteristics. This will enable us to carefully position our work and give the necessary background information to understand the technology used in current virtual lives. We embark on the modelling work of virtual plants, then move on to the modelling of virtual creatures. Both achievements and drawbacks are introduced in each work.

2.2.1 Virtual Plants

As plants are an indispensable part of nature, the computational simulation of plants is also an essential topic in the study of virtual lives. The methodology used to model living plants should take many important factors into account, such as the growth, dynamics and ageing of plants, and external factors like soil, water flow, fertilisers and even the damage caused by insects to the plants.

The first computer model of tree structure is thought to have been constructed by Honda [71]. This model used parameters to define the skeleton of a tree. Aono and Kunii then proposed a three-dimensional geometric model containing a set of rules [7]. By utilising some nonuniform deviators (i.e., attractors and/or inhibitors), this model could bend the branches at particular growth levels without affecting others. Oppenheimer formed trees following fractal theory [117]. He presented a fractal tree model and specified parameters at each branch, including branching angle, the size ratio between the main stem and branch, and the number of branches per stem. De Reffye et al. had impressive results on modelling plants with a procedural model [131]. With the integration of time, their model could grow to a certain age by using probabilities of death, pause, ramification and reiteration. All of this research work has produced considerably realistic plant images, however, the simulation of the dynamic characteristics of plants and their interaction with the external world were not a focal point of these methods.

2.2.1.1 Prusinkiewicz

The pioneering work on modelling plants with "life" properties has been conducted by Prusinkiewicz et al. Their work utilised extended string rewriting systems (L-systems) to describe plants as configurations of modules in space [127]. Therefore, the essence of plant development at the modular level could be conveniently captured by an L-system that replaced individual parent, mother or ancestor modules by configurations of child, daughter or descendant modules. An L-system based virtual plant began with an initial string called the axiom, and proceeded in a sequence of discrete derivation steps. In each step, a set of rewrite rules defined how to substitute modules in the predecessor string by successor

modules. The applicability of a rewrite rule might depend on a predecessor's context (in context-sensitive L-systems), values of parameters (in parametric L-systems), or on random factors (in stochastic L-systems). The resultant strings were interpreted as geometric commands that manoeuvred a LOGO-style turtle [128] in three-dimensions. As a consequence, the development of virtual plants could be presented geometrically from birth to death.

Recent work by Prusinkiewicz et al. has taken the effect of environments into consideration when they modelled virtual plants [101]. In this study, plants and environments were treated as two separate processes, communicating via a standard interface. An open L-system embedded with communication modules was introduced to specify plant models that could exchange information with environments. This study has been successfully applied to capture collisions between plant branches, the propagation of clonal plants, the development of roots in soil, and the development of tree crowns competing for light.

An important issue in Prusinkiewicz's virtual plants is that they have only limited developmental and interactive characteristics, most of which are generated by prescribed modules and rewrite rules. It would be desirable to design virtual plants that can infer themselves growth rules, response rules, and interaction rules with the environment and other living organisms, and have them be changeable with internal/external conditions. Another problem in the modelled virtual plants is that an abstract mathematical formulation of L-systems is absent, which would make their construction more powerful. However, as Prusinkiewicz has said, the "construction of such a theory still seems remote" [128]. It is mainly because of the "lack of a precise mathematical description of plant form".

2.2.2 Virtual Creatures

There have already been a number of impressive efforts in the area of modelling creatures in a virtual world. Here we highlight a few of the most important examples.

2.2.2.1 Boids

The fundamental work on behavioural animation has been done by Reynolds [133], who modelled flocks of birds and schools of fish by specifying the behaviours of the individual animals that made up the group. The aggregate motion of the simulated flock was the result of the dense interaction of the relatively simple behaviours of the individual simulated birds. Each individual in the flock, named a boid, was a point mass attached with a local 3D coordinate system and aligned with its velocity. Based on its local perception of the environment, a boid's behavioural controller computed a steering force at each iteration step of the simulation through arbitration among a series of possible behavioural desires: collision avoidance with nearby boids, velocity match with nearby boids, staying close to nearby boids, and attempting to fly towards a goal. These desires were expressed as acceleration requests that were computed and added in a priority order until the magnitude of the resulting acceleration exceeded the maximum allowed acceleration. Two types of collision avoidance approaches were implemented: one was based on the force field concept which postulated a field of repulsion force emanating from the obstacle out into space; the other was called steer-to-avoid by which the boid found the silhouette edge of the obstacles closest to the point of the eventual impact, and aimed itself to pass by the edge with a suitable tolerance. The former approach may work in undemanding situations, but the latter is a better simulation of a natural bird guided by vision.

The boid animation designed by Reynolds is *per se* superior to the traditional animation techniques that require detailed pose specifications. In each simulation run of the boid flock, the animator only provided the initial parameters of the boid model (e.g., initial position, heading, velocity, etc), and all other aspects of the flock would be implemented automatically and deterministically. The same algorithm of Reynolds' boid model was used to generate some behaviours of the bats in the movie *Batman II*.

Generally speaking, Reynolds' work only simulated the very simplistic, isolated behaviours of low complexity. His boids had simple desires to avoid collisions or fly to a given point in space according to confined principles. Therefore,

these boids had similar behaviours almost everywhere. In addition, the behaviour systems of the boids were specific to little birds or fish, rather than a general behavioural model for many virtual creatures. Sometimes very careful tuning of the low-level parameters were required in order to generate specific behaviours. Nevertheless, Reynolds' original work represents a good example of blending low-level behaviours to arrive at an interesting aggregate motion, and opened the minds of researchers to generating animations by building behaviour controllers.

2.2.2.2 Artificial Fish

The impressive artificial fish created by Terzopoulos et al. [64, 162] have shared with Reynolds' original work a skillful manual design of physical morphology and behavioural control mechanism. Unlike Reynolds' boids, the artificial fish had a fairly complicated physics-based model that consisted of 23 nodal point masses and 91 springs [162]. Moreover, the artificial fish had a more complex behaviour system than that of boids, and could conduct a number of more complex activities, including mating, feeding, learning and predation.

The behaviour system of the artificial fish mediated between its perception system and its motor system. Synthetic vision was adopted and focused on only some of the most useful information for motion. Based on the incoming sensing information, the fish's habit and the fish's mental states (i.e., hunger, libido and fear), an intention generator generated dynamic goals for the fish, such as to hunt and feed on prey. This generator also chose behaviour routines for accomplishing the goals according to a fixed and prioritised set of rules, and the behaviour routines in turn drove appropriate motor controllers to control the simulated muscles of the fish. The artificial fish possessed eight behaviour routines: avoiding-static-obstacle, avoiding-fish, eating-food, mating, leaving, wandering, escaping and schooling. To avoid dithering, the intention generator employed a simple memory mechanism that could "remember" one interrupted intention at a time.

In the early design of the artificial fish, their behaviour control was carefully crafted by using knowledge gleaned from the biomechanics literature. Later, some

learning techniques were introduced to train the fish to learn locomotion skills [64]. Simulated annealing, a stochastic optimisation algorithm, was used to set the parameters controlling muscular coordination of the artificial fish to produce efficient swimming. By applying a short-time Fourier analysis, the learning process then abstracted control functions that produced efficient swimming into a compact representation. Simulated annealing was finally applied again to optimise over the selection, ordering and duration of abstracted controllers so that the artificial fish could perform some compound skills like the stunts a dolphin performs in marine animal parks.

Terzopoulos' fish have achieved striking visual accuracy and realistic looking motion. Nevertheless, the large degree of artificial design by the animator restricted the fish's autonomy and its application to more dynamic and complicated environments. The fixed rules of the intention generator, for example, determined that avoiding obstacles always dominated avoiding predators – this may cause "fatal" problems to the artificial fish when they are much nearer to a predator than an obstacle. When training the fish to perform stunts, the animator needed to introduce additional "style" terms into the objective function of the learning process so as to afford extra control of the learning. As the number of tasks grows, the very delicate human design of such appropriate learning guidance would be very difficult and the optimisation complexity of learning would be very high. Furthermore, learning by simulated annealing is usually too time expensive to be made to work in real time animation.

2.2.2.3 Synthetic Dog

Faced with the difficulties of designing real-time and natural interactions, several researchers have drawn inspiration from other disciplines, such as biology. Blumberg, one of those pioneer researchers, has developed a behavioural control mechanism inspired by findings in ethology (the study of animal behaviours), including behaviour hierarchies, releasers, and fatigue [20, 21]. These findings were used to control a synthetic dog which could interact with others and with the user in a 3D software environment.

The synthetic dog used synthetic vision to extract useful information (e.g. an object, the user's gesture, etc) from an image rendered from the creature's viewpoint. The basic structure of the creature consisted of three parts (*Geometry*, *Motor Skills* and *Behaviour System*) with two layers of abstraction between these parts (*Controller* and *Degrees of Freedom*). The abstraction layers provided a common interface between the three basic parts. As a consequence, the layered behaviour control architecture could be generalisable and extensible.

The purpose of the Behaviour System of the synthetic dog was to determine the "right" set of control signals to send to the Motor Skills system, to best satisfy the dog's goal. Groups of behaviours competed for control of the creature according to their self-assessed values and their mutual inhibition via the "avalanche effect" [107]. That is, the competition was conducted from behaviour groups to subgroups and so on, until a leaf behaviour was selected. Every behaviour was responsible for assessing its own current relevance or value based on its *Releasing Mechanisms* (objects to filter sensory input and identify objects and/or events relevant to the Behaviour), its *Level of Interest*, and the *Internal Variables* (strength of motivations). The winning behaviour sent its commands directly to the motor systems, nevertheless, those behaviours that lost out in competition were still able to express their preferences or suggestions for actions as Secondary commands or Meta-commands. Due to the usage of mutual inhibition and the level of interest of behaviours, the chosen activities neither dithered among multiple activities nor persisted too long in a single activity. They were capable of interrupting a given activity if a more pressing need or an unforeseen opportunity arose.

In Blumberg's behaviour control mechanism, the motor system was hard-wired and learning was not yet integrated into the motor skills generation. The only learning happened in the Behaviour System in which temporal-difference reinforcement learning [151] was adopted to learn instrumental conditioning and classical conditioning action selections. Therefore, the synthetic dog was able to learn how to apply its behaviours in different contexts so as to satisfy previously unassociated motivational variables and how to better predict known contexts.

Motivational variables acted as the reinforcement variable that drove learning. When the motivational variables underwent a significant change, the dog sought to explain the change in terms of its recent behaviours and the recently changed objects in the environment.

The behaviour control mechanism proposed by Blumberg has given us a good example of how ideas from ethology may offer valuable contributions to the construction of interactive and intelligent virtual creatures. The advantages of this mechanism brought by ethology were embodied in relevance (do the right thing), coherence (show the right amount of persistence), and extensibility of behaviours of virtual creatures. Nevertheless, full design of every behaviour and their competition were required for generating correct actions in the synthetic dog. Since the motor system was fixed, the dog had strict behaviours that could not be improved or extended in new situations or for accomplishing new motivations. The only learning in the Behaviour system resulted in a combination of existing behaviours (like the learning of artificial fish), rather than learning of the behaviours themselves. In addition, because the behaviour model of the synthetic dog was built upon the study of ethology, which is more concerned with an explanation at the behaviour level (i.e., what the behaviours are and how they interact), the inherent mechanisms behind behaviours (i.e., how behaviours are generated at the neural level) are unclear.

2.2.2.4 Virtual Humans

In Switzerland, the Computer Graphics Lab directed by Thalmann has endeavoured to model and animate the most complex living system — human being [154]. This lab has created a group of virtual humans that have impressive human appearances and behaviours. These virtual humans used articulated structures as the graphical and animation model. The animation was based on several integrated methods and their blending: keyframe, inverse kinematics, direct/inverse dynamics, and biomechanics-based animation. The virtual humans were aware of an L-system based virtual environment through different kinds of simulated sensors, including synthetic vision, virtual audition and virtual tactile sensors.

Guided by their perceptions, virtual humans could perform a series of human behaviours such as visually directed navigation, tennis playing, and communication with other virtual humans and even the real human user.

Most of the behaviours of virtual humans were simply rule-based, but a few of them had certain adaptive ability. In visually directed navigation, for instance, the navigation tasks were decomposed into global navigation and local navigation [116]. In global navigation, a dynamic octree served as global 3D visual memory and allowed a virtual human to memorise the environment that he "saw". Based on the octree, heuristic search algorithms were used to search possible routes to allow the virtual human to reach global goals. The local navigation strategies were implemented as a series of Displacement Local Automata (DLA), which executed prescribed movements, such as follow_the_corridor, avoid_obstacle, closest_to_goal, and so on. In tennis playing, a specialised "tennis play" automaton was applied in virtual humans to control the tracking of the tennis ball by the vision system. It estimated the future racket-ball collision position and time to perform a hit with given force and a given resulting ball direction. In the communication between virtual humans, information defining the behaviours in one virtual human was passed directly to a second virtual human. Therefore, a virtual human could easily know other virtual humans' postures and then decide corresponding communication actions according to its attitude, character, emotional states, relationships with the other virtual human, and its desire to communicate. During the communication between virtual humans and real humans, a virtual human used an image processing program to match a real human's posture to one of the predefined postures repertoire. Recent studies of virtual humans included the application of virtual humans in network games and group behaviours.

The virtual humans constructed by Thalmann et al. have had very realistic visual effects. However, their behaviours were quite mechanical. Most behaviours of the virtual humans were prescribed. Although some learning and planning were introduced into certain behaviours, they were usually very limited and hence could not be applied to dynamic or unknown environments. In addition, those mechanical behaviours always resulted in the same or very similar behaviours in

different situations and even on different virtual humans. Sometimes, they could generate the always correct behaviours (e.g., in the simulation of tennis playing), and this is almost impossible in reality. Another problem in virtual humans is that they are lacking an integrated, systematic behaviour control model. The behaviours of the virtual humans were studied and implemented from various aspects and they were relatively independent of each other. In order to simulate the congruous, sophisticated behaviours of human beings, a virtual human needs a comprehensive behavioural model that can generate behaviours systematically and with powerful autonomous and adaptive abilities.

2.2.2.5 Woggles

Bates et al., studied the creation of believable agents in the Oz project that proposed to construct artistically effective simulated worlds [12, 13]. Bates et al. believed that one way to create such agents was to give them a broad set of integrated capabilities, even if some of the capabilities were somewhat shallow. Therefore, Bates' agents, called Woggles, had simple reactive behaviours, emotions, and intentionality. The action system for generating reactive behaviours was a rule-based, goal-directed architecture. It had no planning, learning and almost no world modelling. Instead, it used a minimalist conception of goals to manipulate a set of behaviours.

Woggles actually placed great emphasis on conveying their emotional states and intentionality through movement. The goals, and the agents' appraisals of events with respect to the goals, were key to producing a clearly defined emotional state in the creature. Each emotion was mapped into a behavioural feature in a personality-specific way, which in turn affected the action-generating rules of the behaviour architecture.

Although Woggles have exhibited plenty of emotional changes, most of their behaviours and emotional states were artificially designed and very limited. The world that the Woggles inhabited was also constrained to a certain degree. In order to help the user to imagine that the simulated world is real, Woggles may need have not only broad capabilities, but also deep capability in each component.

2.2.2.6 Evolutionary Virtual Creatures

Worthy of mention is the study of Sims on evolutionary virtual creatures whose morphology and behaviour were both generated by using genetic algorithms [148, 149]. This study is different from previous work in which the control system for behaviour is generated for fixed, user-defined structures. The genotypes of the virtual creatures were structured as directed graphs of nodes and connections. Different fitness evaluation functions were used to rate the creatures' capabilities for generating some specific behaviours including swimming, walking, jumping, following, and competing with another creature for resources. As the virtual creatures were evolved towards certain goals, their morphology and behaviour showed conspicuous autonomy and flexibility, although these creatures were only composed of simple cubes.

The evolutionary virtual creatures with simple shapes focused more on evolutionary procedures than the simulation of real creatures. Apart from ensuring that some constraints are fulfilled, the evolution may evolve in its own way and result in interesting but unreasonable creatures. Therefore, the user or human designer has virtually no control over what is going on in the evolution of the virtual creatures. Besides, the evolution of those evolutionary creatures is also the most time consuming process. For example, an evolution with population size 300, run for 100 generations, may take around three hours to complete on a parallel Connection Machine CM-5 with 32 processors. The long time evolution and the lack of control make those virtual creatures unsuitable to live in real-time, interactive virtual environments, at least until processors speed up more.

2.2.2.7 Commercial Products

Apart from academic research work in virtual lives, there have also been some commercial products completed in the entertainment industry. Typical examples are *Creatures* produced by Millennium Interactive Ltd [41], *SimLife* and *El-Fish* by Maxis [100], *Dogz* and *Catz* by PFMagic Inc. [123], *Fin-Fin* by Fujitsu [53], *Galapagos* by Anark [3], and *Black & White* by [19]. These virtual animals, acting as virtual pets, had lovely shapes and appearances, together with some

immediate actions and/or emotions to respond to the human user. By using some artificial intelligence techniques, which were relatively simpler than those proposed in the academic work, the virtual animals possessed some degree of autonomy, intelligence and interactive ability. *Dogz*, one of the very appealing commercial products, could mature into a full dog over time, be fed by the user, play fetch and tug-of-war, and learn to do tricks in return for rewards. *Black & White*, a state-of-the-art computer game, utilised a standard BDI (Belief, Desire, and Intention) architecture [26, 129] augmented in many ways and could learn to achieve various goals by imitation. However, like many other commercial products, *Black & White* creatures usually had a precomputed plan library in which lists of suitable actions can be chosen. It would be better if the creatures knew how to plan dynamically to satisfy a goal. In order to achieve real-time interaction on personal computers that are popularly used in home entertainment, the virtual worlds and animals in many commercial products were over simplified and looked a bit cartoonish. *Creatures*, for example, inhabited a "two-and-a-half dimensional" world: a 2D platform environment with multiplane depth cueing so that objects can appear, relative to the user, to be in front of or behind each other [58]. Nevertheless, these virtual animals worked rather well as entertainment.

2.2.3 Summary

Modelling dynamic, lifelike behaviours of living organisms by the computer is a recent endeavour, nevertheless, much conspicuous work has been done in this area. In the new approach of behavioural animation, many virtual lives have been constructed, including virtual plants, collective boids (birds), artificial fish, synthetic dog, virtual human, and some other virtual creatures. These virtual lives may have or may not have lifelike abstract body shapes and appearances. But their behaviours all exhibited autonomy to a certain degree, by following some behaviour generation strategies describing what to do in what situations. In particular, the boids created by Reynolds have produced lifelike collective behaviours of small, simple organisms. These behaviours are very difficult or almost impossible to create in the traditional animation approach. The artificial fish and

synthetic dog have been endowed with not only impressive visual appearances, but also certain learning ability of organising behaviours. Some virtual lives, like Woggles, have taken some other characteristics of living organisms, such as emotions, intentionality, and interaction with the external world and/or the human user. These characteristics helped a virtual life to be more friendly and believable.

Although previous work on virtual lives has obtained great achievements, it is far from perfect yet. Each work has many of its own problems, as we mentioned in relevant sections. Generally speaking, the behaviours of those virtual lives are still quite limited to human designed rules, and are not suited to changed environments and applications. Most virtual lives have only repeatable, designed behaviours to certain prescribed stimuli, as in virtual plants, boids, virtual humans, Woggles, and some commercial products. A few virtual lives have learning ability, but this learning is quite limited and usually requires detailed knowledge of learning every single behaviour, such as the learning shown in the artificial fish and the synthetic dog; otherwise, the learning of behaviours is often very hard to control, like the behaviours shown in the evolutionary virtual creatures of Sims. These drawbacks in those virtual lives have resulted in much constrained behaviours. It would be much better if the study of virtual lives could be advanced in some way so that virtual lives can learn themselves how to behave well so as to suit new changing situations and motivations, based on their perceived external environmental information, past experiences and obtained knowledge from other behaviours. The virtual lives constructed in this way will relieve the human designer from delicate design work, and most importantly, enhance the lives' autonomy and adaptation abilities. To generate believable behaviours in virtual lives suitable to various environments and applications, further study of virtual life and appropriate behaviour generation strategies is an immediate requirement.

2.3 Virtual Life

In this section, we state our views on *Virtual Life*, and especially, on the construction of virtual lives in virtual environments. To date, the phrase "virtual

life", together with its derivatives like "virtual plants", "virtual humans", "virtual creatures", and "virtual robots", have been often used to refer to simulated objects in virtual worlds, which have life properties. However, little work has been done to study and explain the concept of Virtual Life systematically. Interestingly, there is not yet a clear definition of "virtual life". So, in this section, we first give a working definition to "virtual life" and then analyse it from a systematic viewpoint. We hope this exploratory work will stimulate more investigation and development of virtual lives.

2.3.1 Definition

Whenever a concept involves abstraction, it is hard to define it with absolute precision. In fact, there is not yet an exact definition of "life". Many biologists try to define "life" by stating what it looks like and what it does. So here we would like to use the same method to define "Virtual Life".

A *Virtual Life* is a computational entity that simulates real lives in virtual environments. It has a lifelike visual shape and appearance, and believable behavioural patterns. Inhabiting a virtual environment, a virtual life can perform actions autonomously, adapt to environmental changes, and interact with the outside, especially with the human user, by characteristic activities.

People may have come across some other terms that are very close to the study of simulation of living things, including "digital actors", "synthetic agents", "softbots", and "avatars". These terms, together with "virtual life", sometimes are used synonymously. However, their origins can be traced back to disparate applications that are different in terms of scope and purpose. In terms of the simulation of real lives in virtual environments, we prefer the term, "virtual life".

2.3.2 Construction of Virtual Life

For a true feeling of *presence*, convincing graphics and believable actions of a virtual life are both important. Realistic action can enhance the realism of graphics, while geometric and texture fidelity can make actions more intriguing. There-

fore, visually realistic shape and appearance and believable behaviours are two important components for constructing virtual lives.

Simulating lifelike visual images and motions is an essential topic in computer graphics. With dramatically improved computation speed and control methods to portray computer graphics, people are no longer satisfied by visually impressive objects with simple or cartoonish actions. However, when a simulated object shows complicated behaviours, it is usually very mechanical and non-interactive and the result of a painstaking human design process [93]. This problem is especially significant in the application of virtual environments, which are highly dynamic and interactive worlds continuously changed by the participation of virtual lives and the human user. Since the user may wish to interact with the virtual environments and their virtual lives at will, it is incredibly hard for the human designer to predict every possible action of every user and program the corresponding reactions into virtual lives beforehand. Ideally, it should be up to the virtual lives themselves to observe and analyse the user's behaviours, and then decide when, where, and how to move its body to react to the user appropriately. Therefore, virtual lives should be self-controlled, self-animated by simulating the natural mechanisms fundamental to life. To enable this, a virtual life may have sensors to accept external information, and effectors to create movement. More importantly, it needs a behaviour model to analyse and process information, make suitable decisions based on its own intentions, and perform varied tasks in a dynamically changing environment without human intervention. Therefore, the design of a virtual life involves a visual model and a behaviour model.

2.3.2.1 Visual Model

To produce lifelike living things with realistic visual effect is always a challenge in computer graphics. Generally, it involves a number of stages:

- Generate three-dimensional models,
- Determine viewing specifications, such as skin texture and hair,

- Calculate colour values of visible surfaces and shadows, and
- Define animated sequences with time-varying changes.

Because of the complexity of the world, it is still impossible to achieve perfect visual realism, especially in a real time simulation. Effective and fast methods are still being researched. In the conflicting requirements of realism and real time, suitable compromises must be made.

2.3.2.2 Behaviour Model

The major interest of this thesis is in the behaviour model of virtual lives. A sound behaviour model is indispensable for a virtual life to produce appropriate behaviours and to maintain a good life in a virtual world. In the study of real lives, many people believe that the essence of life is in its metabolism, reproduction, autonomy, adaptation, responsiveness, etc. [147]. Compared with real lives, we think a virtual life, and particularly its behaviour model should have at least the following characteristics:

1. Autonomy

Autonomy is an important aspect of virtual lives. Indeed, autonomy is universal in natural organisms but not confined to them. It is reflected in the movement that occurs within an organism or results from internal changes [147]. A virtual life is said to be autonomous in the sense that it senses and acts in its environment, and decides itself what actions to take so as to best achieve its goals [94]. Such kinds of behaviours represent the capacity of the life to maintain its viability in varied, changing environments. Plants may seem motionless, but they also experience spontaneous growth toward a position with better living conditions. Although the growth is a slow movement, there is considerable movement within their cells, and from cell to cell.

The need to have autonomous behaviour for virtual lives arises from two considerations: the less work done by the human designer and the more faithful

illusion in the user that a virtual life is a real one [155]. As we mentioned before, due to the dynamic nature of virtual environments and virtual lives, it would be very hard and time-consuming for the human designer to design every detailed behaviour of every virtual life beforehand. However, if a virtual life always stops to seek help from the human designer during its running or just "paralyses" suddenly when it encounters problems, it would cause great disappointment to the user. Therefore, it would be best for a virtual life to possess powerful autonomy so that it can analyse the environment information actively, and decide how to relate its receptor information to effector actions correctly by itself. In particular, when a virtual life is in an urgent, difficult situation, it should be able to find a way to solve its problem promptly and spontaneously without intervention from the designer. To date, many autonomous techniques have been presented in the research area of autonomous agents, robotics, and some other related areas, and resulted in very interesting autonomous behaviours. However, despite these achievements, building full autonomy in virtual lives remains an elusive goal, especially in real-time applications.

2. Adaptation

Another indispensable characteristic of virtual lives is adaptation. Adaptation of a virtual life is the way it is organised to improve itself over time in the environment it inhabits. There is a continuum of ways in which a virtual life can be adaptive, like real lives [94]. In a narrow sense, adaptation is to enhance a virtual life's ability via learning so as to make it survive in more or less unpredictable and dangerous environments. This includes how to utilise environmental resources best and how to change situations for its benefit. In a broad sense, adaptation is to adjust a virtual species genetically to changed environmental conditions. When a species reproduces, it is not just a simple copy, but a complicated transfer of certain parts (structures) to its offspring. This reproduction with changes is responsible for the evolution of life. Although the broad adaptation always takes a very long time in nature, the long time evolution is not necessary

in a simulated world. Virtual evolution in generations may be achieved in a short time in simulation by showing the procedures the human user is interested in only.

With suitable adaptation ability, a virtual life or a virtual life species can learn to improve its behaviours continuously for a better subsistence in its virtual environments.

3. Interaction

In real lives, responsiveness is a distinctive feature too. It occurs to some extent in all living things. Living organisms not only maintain themselves through environmental changes, they also respond to these changes by characteristic activities [147]. Even plants will react to changes in sunshine, nutrition, and other environmental conditions. Responsiveness is extremely obvious in higher animals, e.g., human beings.

Virtual lives should be able to interact with their outside (e.g., the virtual environment, other virtual lives, and the human user), for an exchange of the best "living resources" to maintain in the virtual environment. In particular, virtual lives should be able to interact with the human user who is the main actor in virtual reality. The user in VR will expect to be able to interact with virtual plants, animals, humans, and to see what response he will get [155]. It would be very intriguing if the user receives distinctive responses from virtual lives during their interaction. For example, the user must be very frustrated if he continuously looks at a fossil that has not changed for millions of years. However, the user may be very interested and satisfied, and think that it can "understand" him if the fossil exhibits its evolutionary procedures in a short time and with emphasis on the user's interests. In this regard, virtual lives are not just a simulation of real lives, even though they come from the real ones.

Here, we rename "responsiveness" to "interaction" as a characteristic of virtual lives for coordination with Virtual Reality in which interaction is an essential factor.

The above three characteristics are fundamental to virtual lives. They are the basic requirements of virtual lives in virtual environments. Without any one of them, a virtual life may not "live" well in its environment. That is, the life may not look believable or useful from the human user's point of view. Lives at higher levels will have more features. Virtual humans, for example, still have their own personalities, emotions, high intelligence, and some other characteristics [155]. These extra features will make a high-level life more functional and believable, but they are not universal or indispensable to all of the lives. Rather a virtual life can be profitably endowed with varying levels of both fundamental and extra features depending on the nature of the life and the nature of its environment.

2.3.3 Virtual Life vs. Real Life

As we mentioned before, virtual lives are not a simple copy of the real lives. They derive from the real lives, but have their own features since their living environments are simulated ones.

An essential distinction between virtual lives and real lives is that the metabolism, growth and reproduction, the essential hallmarks of real lives, could be left out in virtual lives. In nature, every organism goes through procedures from birth, development, to death. Some organisms also reproduce their kinds in generations. These procedures are maintained by metabolism. There are also frequent information flows in virtual lives, which are often thought of as the metabolism of virtual lives. However, the life time of a virtual life may not depend on its "metabolism", but its application roles in a virtual environment. A hero in a game, for example, may be mature when he appears for the first time. He may never die as the user wishes. But the virtual dinosaur species may only live for several hours and become extinct quickly. In other words, the standard life period of natural organisms is no longer a necessity to virtual lives.

Another difference between virtual lives and real lives is that a virtual life may possess some abilities and features that its corresponding real life does not have. Some virtual lives may even not exist on the earth. If a user comes into a computer-generated fairy world, for instance, then trees speak, virtual humans fly,

and virtual robots walk around and do their job. Moreover, virtual lives may have overstated body shapes and appearances, and exaggerated behaviours to capture the user's interests. This is a technique commonly used in the animation world. In Disney animation, for example, a character tends to rear back in preparation for a rapid forward motion. Strictly speaking, this behaviour is unrealistic, but it is essential for people to grasp the meaning in the motion. So, being lifelike or "alive" is not the same as being realistic [12]. However, one should bear in mind that, whatever unrealistic changes are made in virtual lives, they should help the user to think those virtual lives are believable and motivate the user to interact with them. It is in this sense that "believable" is used in our definition of *Virtual Life*.

2.4 Our Work

The previous work on behavioural animation has made a breakthrough on giving virtual lives believable behaviours, as we have introduced. However, compared with natural behaviours of living organisms that are autonomous, adaptive and interactive, the overall behaviours generated in behavioural animation still have many deficiencies. In particular, those behaviours have a common but serious problem in that they lack a powerful autonomous and adaptive functionality. This is essentially reflected in that those virtual lives cannot improve their behaviours continuously so as to deal with new complex situations, nor can they be aware of generating new behaviours based on their own knowledge and resources so as to serve new behavioural motivations. In consequence the human design work on virtual life behaviours is usually very high, and the autonomy and adaptation of virtual lives are affected as well.

In Section 2.3.2, we explained that the design of a virtual life includes two important components: a visual model and a behaviour model. In this thesis, we mainly focus on the construction of a reasonable behaviour model as our first step towards the construction of a complete virtual life. The design of a vivid visual model is one of our future goals.

We present a Computational Behaviour Generation (CBG) model in this thesis, which is a natural progression of virtual lives in an attempt to construct a useful behaviour model and suitable behaviour generation strategies for them. The CBG model helps virtual lives build effective adaptation and autonomy in their behaviours based on continuous interaction with the external environments. By using this behaviour model, a virtual life can not only learn to generate behaviours for achieving particular behavioural motivations, but also learn to generate multiple behaviours without human intervention. The learning ability of the CBG model presented in this thesis involves two aspects: learning to improve its behaviours so as to deal with more and more situations (these situations may be unknown to the virtual lives, and much more complicated than any other situations met before), and learning to perform new related behaviours based on pre-learned behaviours by making use of their common knowledge and resources (i.e., multifunctional learning). This learning ability is particularly achieved by a novel learning algorithm, a Multi-agent based Evolutionary artificial Neural network with Lifetime learning (MENL). With this effective learning algorithm, a virtual life is able to decide appropriate motor actions on its own, and adjust to a wide range of situations and behaviours dynamically. As a consequence, the virtual life can obtain certain autonomy and adaptation in its behaviours. The human designer can therefore be relieved from very heavy and time-consuming design work. The adaptive and multifunctional Computational Behaviour Generation model proposed in this thesis is not designed for some specific virtual creatures. Instead, it is intended to be built as general purpose so as to suit many virtual lives.

In addition to the work on the construction of believable behaviours for virtual lives, similar work has also been done in robotics, which pursues animal-like or human-like behaviours in robots inhabiting the real world. Due to the huge amount of literature in this area, we won't review them here, but relevant work will be introduced in specific chapters. In comparison with the research work on both virtual lives and robots, our work on the adaptive and multifunctional Computational Behaviour Generation model has presented many original properties,

which are embodied in its neuroscience inspiration, the hierarchical architecture of the CBG model possessing many potential strengths, a Multi-agent based Evolutionary artificial Neural network with Lifetime learning (MENL) that has robust and continuously improved learning ability, and the multifunctional learning of the CBG with MENL that can efficiently learn multiple behaviours together. These properties will be explained in detail in the following chapters.

2.5 Conclusion

This chapter has presented a snapshot of current development of virtual lives and our attitude towards *Virtual Life*. In particular, we have reviewed recent work on constructing realistic and believable behaviours in virtual lives, and pointed out some major problems in that work. When explaining our views on virtual lives, we presented a working definition of *Virtual Life*, which is currently absent, and some suggestions on constructing virtual lives. We believe a visual model for visually lifelike shapes and appearances and a behaviour model for believable behaviours are two important components of virtual lives. In addition, this behaviour model should have at least the characteristics of autonomy, adaptation and interaction so as to help a virtual life maintain a normal life in changing virtual environments. Finally, we have described how our work on the CBG model and the MENL learning algorithm fits into virtual life research.

Chapter 3

The Computational Behaviour Generation Model

In this chapter, we present the architecture of the adaptive and multifunctional Computational Behaviour Generation (CBG) model for virtual creatures, which is capable of producing various behaviours in various environments and situations. The next section is a brief introduction to the CBG model. Section 3.2 discusses recent work on the construction of behaviour control models for virtual creatures, which is related to the CBG model and places our work in this context. Section 3.3 explains some biological results of neural control of behaviour, which are fundamental to the design of the CBG model. Section 3.4 describes the proposed CBG model and its formal description. Section 3.5 introduces several potential strengths held in the CBG model. Space occupying behaviours, which are functions fundamental to many natural animals, are introduced in Section 3.6. In this thesis, these behaviours are used as model behaviours for the CBG and MENL to learn both individually and multifunctionally. In Section 3.7, we give an instantiation of the CBG model, which shows how the CBG model can be implemented to generate space occupying behaviours. The last section concludes with a summary of the chapter.

3.1 Introduction

As a behaviour generation model, a basic requirement of the CBG model is that it should be able to implement the whole procedure from planning to execution of motor actions for a designated behaviour, by initiating and coordinating suitable motors of virtual creatures. A major objective for the CBG model is that its architecture should allow and encourage adaptive and multifunctional implementation of behaviours. In addition, this architecture should be as general as possible so as to serve a variety of virtual creatures.

The construction of the CBG model is mainly derived from an understanding of the hierarchical behaviour control system in the brain. Similar to the natural behaviour system, the CBG model has *sensors* and *motors* to interact with the outside world, a body *State* module to report body activities of a virtual creature, and a behavioural *Motivation* module to generate behavioural motivations based on body needs. In addition, the CBG model contains within it a *Computational Motor Control (CMC)* system that it utilises to perform decision, programming and execution of motor actions for achieving various behavioural motivations. The CMC system is hierarchically composed of a *Strategy* module, a *Program* module and a *Movement* module. The Strategy module is responsible for designing a general strategy for achieving a particular behavioural motivation. It also selects the appropriate motor actions to execute in the current situation. The motor actions are decomposed into detailed motor programs in the Program module. The actual initiation of motors for executing every motor program is engaged in the Movement module. Based on this hierarchical control, the higher levels (e.g., the Strategy and Program modules) of the CMC system only make general design of motor plans and programs concerning a designated behaviour. It is the lower level (e.g., the Movement module) that implements this design by encoding each element of movement in detail. During behavioural execution, the CBG model monitors body states and environmental information at every step, so every level of motor control can adapt according to both internal and external changes of the body. The architecture of the CBG model designed in this way has provided many potential strengths like adaptation and multifunctionality, as

we will see in Section 3.5.

3.2 Related Work

To design a reasonable behaviour control model is very important to the construction of virtual creatures. This model is directly responsible for generating proper behaviours, and hence maintaining a normal life of virtual creatures in virtual environments. A sensible behaviour control model for virtual creatures should be complete so as to include the whole procedure of selection and execution of motor actions for generating behaviours. It should also be general so as to implement various behaviours on a variety of virtual creatures.

In previous work on virtual creatures, as we reviewed in Section 2.2, several behaviour control models have been proposed for controlling virtual creatures' behaviours. Those models have provided the behaviour control of virtual creatures of many different types and complexities. Some behaviour control models, such as the model of Woggles created by Bates et al. [12, 13] designed certain behaviours composed of a set of unchanging action plans and the implementation of those actions was a direct manipulation of the graphical elements of virtual creatures. The behaviours designed in this manner were heavily combined with the graphical models of virtual creatures. If the graphical models or part of the behaviours were changed, the generation of the entire behaviours had to be redesigned. The incapability of these behaviour control models to respond to changes restricted them to only a few virtual creatures that have prescribed behaviours.

Some research work on virtual creatures has concentrated on designing elaborated motor movements. The resulting virtual creatures, such as the virtual humans created by Thalmann et al. [154] had delicate body shapes and structures, and involved lots of body joints. In order to produce an effect of natural-looking motions, biomechanical data and control techniques were often used to coordinate the sophisticated joint and muscular activities of those virtual creatures. However, in these virtual creatures with visually lifelike motions, the information about when and what motions to produce was often *a priori* built into them,

rather than arising from the autonomous decisions made by the creatures themselves. In contrast with their realistic movements, these virtual creatures lacked the powerful strategies that could decide actions autonomously and adaptively.

The behaviour control model of artificial fish constructed by Terzopoulos et al. [64, 162] was an integration of both selection and execution of motor actions. This model organised these two abilities as two layered individual modules. As the top-level module of decision-making chose suitable motor actions and calculated necessary movement parameters, the low-level module transferred the parameters into exact graphical values and initiated corresponding graphical elements (motors) to move. Behaviour control models organised in this manner separated motor action selection from real graphical models of virtual creatures. As a consequence, they could be used to produce more complicated, compound actions that are composed of a series of motor movements. However, in artificial fish, the movement parameters generated by the high-level module of motor action selection were embedded with much information specific to fish movement, the behaviour control model was therefore limited to a particular species (fish) that had particular motors (fish-like muscles).

Blumberg [20, 21] has applied another kind of hierarchical behaviour control model in his virtual creatures, which had two modules named *Controller* and *Degree of Freedom* (DOF) working as abstract barriers between behaviours and motor skills and between compound motor skills and the geometry of virtual creatures. The Controller mapped generic commands of behaviours into motor skills and relevant parameters. The DOF then mapped abstract movement parameters (often numbers between 0 and 1) into the graphical space of the creatures via interpolators and inverse kinematics. Due to these two abstract barriers, the input commands sent from the higher levels were relatively independent of the concrete virtual creature and its graphical model. The behaviour control was hence general to a series of virtual creatures with different motors.

Along with behaviour control models constructed for virtual creatures, a large number of behaviour models with partial or complete behaviour generation procedures have also been presented in the robotics area [1, 2, 27, 34, 67, 135, 161].

In particular, [1, 2, 135] proposed robot control models sharing similarities with the layered architecture of the vertebrate brain. Our proposed Computational Behaviour Generation (CBG) model is similar to these models and the one of Blumberg's, in the sense that it is also a general and complete behaviour control model. The CBG model generates control information not only for selection of motor actions (in the Strategy module), but also for programming and execution of these motor actions (in the Program and Movement modules respectively). The selection of motor actions can therefore be based on abstract and general motor actions, since the interpretation of motor actions into detailed motor programs and then real movement signals is implemented in the Program and Movement modules respectively.

Compared with other behaviour models in both virtual creature and robot studies, our CBG model has also been designed with important information and mechanisms for adaptive and multifunctional implementation of behaviours. In particular, the bidirectional feedforward and feedback information flows in the natural behaviour control have been revealed and implemented in the CBG model. Due to the bidirectional information flows, the CBG model is able to obtain information of both top-down control and bottom-up sensory feedback so continuous adjustment of behaviours can be made according to both useful and harmful experience. The resulting behaviours can therefore have improved functionality in a wide range of situations. The adaptive learning algorithm adopted in the CBG model, MENL, has also shown its originality and advantages, when compared with other adaptive learning technologies, such as evolutionary robotics and reinforcement learning. This will be introduced in detail in the following chapter. Another conspicuous property held in the CBG model is its multifunctionality. The multifunctionality suggests that the CBG model may be able to use the same mechanisms to generate a series of functionally related behaviours by taking advantage of the shared resources and knowledge across these behaviours. Because the CBG model is constructed in a hierarchical and general way, many components at different levels are reusable for different behaviours. To implement multifunctionality in the same behaviour control mechanisms is a concept

broadly overlooked in previous behaviour control models for virtual creatures and robots. Those models usually cannot use what they have obtained for one behaviour to generate other relevant behaviours in the same control mechanisms. Having strong adaptation and multifunctionality, the CBG model can produce multiple behaviours with varied motivations in varied conditions. In consequence the CBG model can reduce the laborious and difficult work of artificial design. The experimental results in the following chapters have supported this.

In the next section, we will briefly introduce the recent studies on the neural control of behaviour, which provide the fundamental basis of the CBG design.

3.3 Biological Basis

The gross features of the natural behaviour control system in the brain (especially the vertebrate's brain) provides a good example when we design an artificial behaviour controller for virtual creatures. In particular, the hierarchical structure of the natural behaviour control system, the input from body states and environmental information, the adaptation by learning, and the generality in various behaviour systems gave us direct and useful ideas when we designed the architecture and functions of the CBG model. In this section, we introduce the characteristics of the natural behaviour control system briefly. For detailed explanation of the natural behaviour control system, see [79, 29].

3.3.1 Hierarchical Behaviour Control System

The pathways between sensory input and motor output are a complex system in both anatomical and functional aspects [6]. There are a number of different structures intervening between sensors and motors, and there is a great deal of interaction between these structures. However, a voluntary decision regarding which action to perform is usually controlled by roughly hierarchical levels in the brain [29, 79, 122, 142]. This hierarchy is illustrated in Figure 3.1, a diagram of major components involved in voluntary movements.

In the hierarchy shown in Figure 3.1, the limbic system governs basic bio-

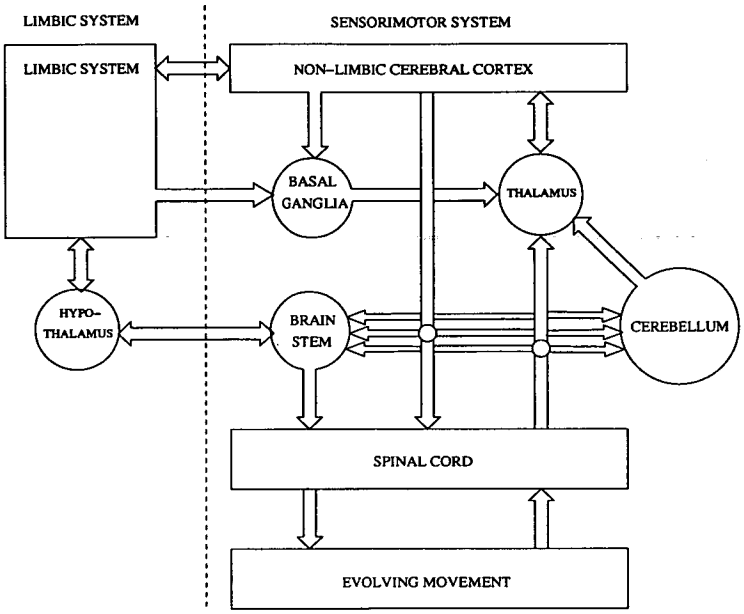


Figure 3.1: Diagram of major components involved in voluntary movements (after [29]).

logical drives of the body, including feeding, drinking, reproduction, and other life-preserving activities. These drives are regulated by the hypothalamus. After the limbic system makes decisions on the relevant information about the perceived needs of the body, the sensorimotor system is then enabled to convert needs into actions.

The sensorimotor system deals with (nonlimbic) sensations, their perceptions and sensorimotor functions. The highest level of the sensorimotor system, the nonlimbic cerebral cortex, is responsible for elaborating perceptions and forming overall motor plans (strategies) for the body needs. The strategies are converted into detailed motor programs (tactics) at the middle level of the sensorimotor system, which consists of the brain stem, the thalamus, the cerebellum, and part of the basal ganglia. Motor programs at this level determine and correlate move programs and hold programs, such as body equilibrium, movement directions, force, speed, and mechanical stiffness of the joints. The tactical instructions from

the middle level are then carried by descending paths to the spinal cord, the lowest level of the sensorimotor system. In the spinal cord, those instructions are finally coordinated and translated into exact joint and muscular activities that are regulated through stretch reflexes.

Through the above description of the hierarchical behaviour control system, we see that the higher levels of the control system are responsible for collecting all the information relating to a behaviour to implemented, and making general action plans for carrying it out. Instead of controlling each basic element at the lower levels directly, those action plans are only general commands that are sent to relatively autonomous lower level systems. The lower level systems then analyse those general action plans, produce basic spatiotemporal patterns underlying movements, and eventually drive concrete motors to move. This kind of hierarchical control information flow from general to specific relieves higher levels from having to consider every exact property of lower levels, and simplifies the control of one level by another.

3.3.2 Body States and Environmental Information

The hierarchical control system of behaviours correlates and adjusts its actions by means of two types of signal: internal body states and external world information. A *body state* is the spatial distribution of body activities at a given moment in time [56]. It may involve the configuration of the body (such as joint angles, angular velocities, and muscle positions and load), the interaction of the body with the world (such as contact with an object), and even the neuron activities in a microscopic view (such as neuron membrane potential, firing frequency, the onset and termination of bursts, the integrated activity across a population, and the like). When the body is situated in an environment, environmental information, such as the location of the object to be manipulated, the available places and obstacles in the environment, etc., is another essential factor to affect motor actions. Body state variables and environmental knowledge can be collected from sensory signals and information sent from related regions in the brain, or estimated when exact information about them cannot be acquired directly. The

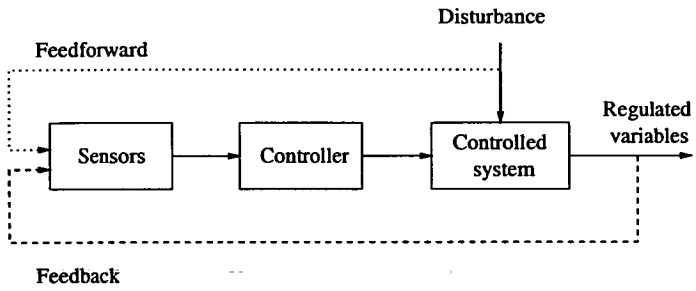


Figure 3.2: Feedforward and feedback configurations for regulation of a controlled system (after [72]). The system is feedforward (open-loop) when the regulation by the controller is only based on sensors that detect potential disturbances. The system is feedback (closed-loop) when effects of disturbances on regulated variables are considered.

information about the body states and the environment provides the behaviour control system with fundamental knowledge about the body itself, both internally and externally. This information can contribute to the preparation of movement, and more often, it can help the adjustment and execution of the overall plan for generating a behaviour, by informing the behaviour control system of instant changes of the body and environment after every action execution. The responses of the body and environment to executed actions constitute the important feedback to the behaviour control system (Figure 3.2) for adaptive learning, as explained in the next section. In a purely feedback-controlled system, the only information that needs to be stored in the behaviour control system is the body state variables or behaviour objectives since it is the environment that holds the information necessary to control movement. Taken together, body needs, body states, and environmental information are sufficient for the behaviour control system to determine future movement of the body.

3.3.3 Learning

Behaviours are adaptive with learning [29]. Learning can help behaviours to adjust to a wide range of situations and applications.

Learning happens when behaviours are not executed as expected. Once a behaviour is launched, the behaviour control system monitors its progress via continuous sensory feedback including changed body states and environmental responses to executed movements as described above. In this closed-loop system (Figure 3.2), the sensory feedback reflecting the movement progress is sent back to all levels of the sensorimotor system for a comparison with the commands previously issued for intended movement. Any difference between intended and actually executed movement will be found out, and the proposed action plan can be adjusted accordingly. Inappropriate commands should be removed from or revised in the plan, as correct commands are kept for the next use. When the behaviour has been repeated many times, the behaviour control system can learn a proper action plan and a correct execution of these actions. The behaviour can therefore be produced smoothly and successfully thereafter. Sometimes, when speed is the critical requirement or when immediate sensory feedback is unavailable, behaviours may be carried out straightforwardly without instant feedback, but the proposed action plans can still be corrected at the end of behaviour execution, if the final behaviour results are not correct. The complete ignorance of feedback may apply to very well learned behaviours [29]. In this case, a mode of "open loop" control (Figure 3.2) is formed (i.e. feedforward only).

Behaviours can also be learned from past experience when they are successfully produced. When success has been achieved for a behaviour, it will be recognised at the higher levels of the behaviour control system, and sent back to lower levels successively. Both neural and muscular activities associated with the behaviour are committed to memory as programs. Repeated use of the same programs increases the accuracy of the memory, and hence that of the behaviour generation. Thereafter, the successful behaviour can be executed more automatically to a greater or lesser degree. This kind of learning from success is representative of many animals [29].

By learning from both harmful and useful experience, behaviours are continuously improved in performance.

3.3.4 Generality in Behaviour Systems

A *behaviour system* describes the assembly of elements that is most actively involved in a particular behaviour at any given time [29]. Although different behaviours may differ in their nature, their drives, the experience of the subject, and some other conditions, they have similar overall information flows in their execution. For example, a general scheme concerning how to perform a behaviour usually comes first at the highest level of the control system, which is then decomposed into motor programs and motor commands suitable for concrete motors to execute at lower levels. At the same time, all levels of the system are informed about successful or unsuccessful execution of the behaviour for learning and adjustment. The whole control system is a mixture of the top-down control information and bottom-up sensory feedback.

By making general decisions first and leaving implementation details to be decided later, the natural behaviour control system also provides many shared components at various levels in different but functionally related behaviour systems [29]. If we have learned how to use a pen to write on paper, for instance, we will very probably know how to use a brush to write on a big board. The written letters in those two approaches are usually very similar. In this case, the same general writing scheme is implemented on different muscles. The same set of motors can also be used for different behaviours in some other cases. Knowing that we use the same arms to perform different behaviours such as swimming and writing, it is clear that sometimes we have to use our limited motors to execute numerous behaviours although the general strategies of these behaviours are different. Because the high level strategies do not manipulate the detailed motor execution directly, it is the lower level systems that determine what motors to use from a limited set of motors.

As the natural behaviour control system produces varied behaviours, it is important that the common components involved should be able to make the right decisions for those behaviours at the right time. Many neural networks in the natural behaviour control system can be reconfigured to provide different behaviour outputs. These neural networks are called *multifunctional neural networks* [55].

In the nudibranch mollusc *Tritonia*, one set of premotor interneurons has been found to be able to generate the pattern for swimming at one time and the pattern for withdrawal at another time [55]. Since then, many multifunctional neural networks have been found in many animals. Due to the hierarchical structure of the behaviour control system and multifunctional neural networks, varied behaviour systems share their generality in not only the mixed information flows, but also many common units across them.

3.4 The Computational Behaviour Generation Model

From the above description of the biological behaviour control system, we know that:

- The formation of a behaviour is initialised in the motivational limbic system, and designed, programmed and executed in the hierarchical sensorimotor system, with the assistance of body states and world information;
- Behaviours can learn from their successful or unsuccessful experience with the help of sensory feedback;
- Different behaviour systems have similar feedforward and feedback information flows during behaviour implementation, and functionally related behaviour systems may have the same functional units to share.

Inspired by these biological concepts, we present a Computational Behaviour Generation (CBG) model that can produce complete behaviour control for virtual creatures, with concurrent adaptive and multifunctional strengths. In addition, the CBG is designed with the consideration of easy and effective implementation on a wide range of virtual creatures. Nevertheless, it should be noted that, rather than mimicking the natural behaviour control and learning system exactly, the CBG model and the following MENL learning algorithm proposed in this thesis only take the spirit of the natural system for reference. The real natural behaviour control system is indeed a much more complicated mechanism involving numerous interrelated structures.

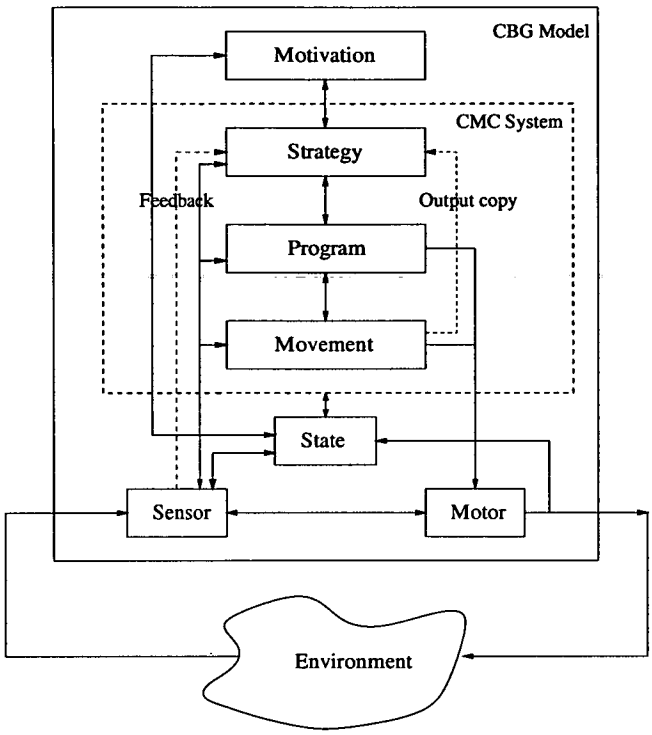


Figure 3.3: The basic structure of the Computational Behaviour Generation (CBG) Model

3.4.1 The Basic Structure of the CBG Model

The basic structure of the Computational Behaviour Generation model is shown in Figure 3.3. Similar to the natural behaviour control system, the CBG model consists of a Motivation module, a hierarchical Computational Motor Control (CMC) system, a State module, and Sensor and Motor systems. All parts of the CBG model cooperate and coordinate with each other to generate behaviours.

CBG sensors and motors are the interface of the CBG to its external world. The former (sensors) are responsible for collecting external environmental information for the CBG, including perceived environmental knowledge and environmental responses to body movements of a virtual creature. The latter (motors) are the concrete mechanism to perform motor actions commanded by the Computational Motor Control system. Because virtual creatures are actually graphical models in virtual environments, their sensors and motors are also simulated ones.

Particularly, CBG motors are procedures that attempt to modify graphical entities embedded in the graphical model of a virtual creature for producing a sequence of animated motions.

The State module in the CBG model records the current and previous body state information (e.g., motor positions, activities of all levels of the CMC system, etc.). This information provides the fundamental basis for selecting, programming and executing motor actions. The State module is monitored by all levels of the CBG, so the exact movement required for carrying out a behaviour is dependent on the states of the body.

The Motivation module is the command system of the CBG model. It collects desires or needs of a virtual creature and generates corresponding behavioural motivations for the Computational Motor Control system to achieve. If there is no obvious body desire presented or if the Motivation module falls "asleep" after initiating the CMC system, the CBG model may still work well, but in a less purposeful manner. A typical behaviour example of this type is wandering, a kind of movement from place to place without any special purpose or destination.

The Computational Motor Control (CMC) system is the core of the CBG model. It is responsible for recognising, selecting and analysing behavioural motivations, and implementing them based on relevant sensory information. The CMC system of the CBG model is hierarchically constructed by three separate but interrelated modules: Strategy, Program and Movement. The demands expressed by the Motivation system are first analysed by the Strategy module, and in there, general strategies for fulfilling these demands are formed. The Strategy module also makes decisions about what motor actions to take in the current situation, and the decisions are passed to the Program module. The Program module then converts abstract decisions into more detailed motor programs that are appropriate for motors to execute. Because body movements, which always involve several joints, can be composed of simple movements that affect only one joint [29], it is in the Movement module that those basic and simple motor movements are formed. The simple movements are subprograms of the motor programs designed in the Program module. So, when the Program module informs the

Movement module when and which specific motor action to take, the Movement module will translate these commands into specific simple motor movements.

Similar to the biological behaviour control system, the hierarchical control of the CMC system is assisted by the State module and the external world. For selecting and executing appropriate motor actions, the CMC system is continuously receiving with body information recorded by the State module and environmental information collected by the Sensor system. The concrete execution of motor actions by the Motor system in turn updates State variables and environmental information, which are necessary for correct selection and execution of motor actions at the next step. Body and environmental feedback on executed motor actions are sensed by the Sensory system and sent to all levels of the CMC system, especially to the Strategy module, the highest level of the CMC system. The Strategy module already knows about what motor actions are to be executed via output copies of Movement commands (see Figure 3.3). Thus, through examination of the sensory feedback, the Strategy module can have an image of how the intended actions have been performed. By comparing the intended actions with their completion, the Strategy module is able to adjust its proposed action plans accordingly so that a proper generation of a designated behaviour can be effected.

After each motor action execution, the CBG model will check whether the desired target state for a behaviour is reached. If the target is reached, this means the proposed motivation is satisfied. Otherwise, the distance between the present state and the target state will continuously drive the CBG model to completion.

3.4.2 Formal Description of the CBG Model

The Computational Behaviour Generation model and its behaviours can be described as a tuple $\Sigma = (\mathcal{S}, \mathcal{E}, \mathcal{G}, \mathcal{A}, \pi, \psi)$, where

- $\mathcal{S}$ is a nonempty set of body states,
- $\mathcal{E}$ is a set of environmental information, including environmental knowledge

perceived by the virtual creature and environmental responses to executed motor actions,

- $\mathcal{G}$ is a set of motivational goals (it can be null in some cases),
- $\mathcal{A}$ is a nonempty set of motor actions,
- $\pi(\mathcal{S}, \mathcal{E}, \mathcal{G}) \rightarrow \mathcal{A}$, is the action function mapping each input triplet $(\mathcal{S}, \mathcal{E}, \mathcal{G})$ into an action in $\mathcal{A}$, and
- $\psi(\mathcal{S}, \mathcal{A}, \mathcal{E}) \rightarrow \mathcal{S}$, is the state transition function.

A virtual creature may have finite motor actions $\mathcal{A}$, body states $\mathcal{S}$, and motivational goals $\mathcal{G}$. However the environments it encounters may be endless, so the environmental information $\mathcal{E}$ can be infinite. The state transition function ψ can be either deterministic or non-deterministic, depending on the nature of the performed behaviours and related actions.

Assume at time t, the body is in state s_t, at situation e_t, and driven by motivation g. Then the action $a_t \in \mathcal{A}$ selected for execution at time t is

$$a_t = \pi(e_t, s_t, g) \tag{3.1}$$

After the execution of a_t, the body state is consequently changed to s_{t+1} at time $t + 1$. According to the state transition function, it is known that

$$s_{t+1} = \psi(s_t, a_t, e_t) \tag{3.2}$$

3.4.3 State Transition Diagram

A virtual creature usually has many body states. These states result from various motor actions taken by the creature. According to actions performed and their contributions to a designated behaviour, body states of a virtual creature can be functionally mapped into several categories. The relationship between motor actions and changed body states constitutes a kind of state transition diagram that can explain how behaviours are executed.

In the case of space occupying movement in environments for certain motivations, for instance, the body states of a virtual creature can be simply classified into five major state categories: No-action, Turn, Movement, Collision and Target. These states are caused by different kinds of motor actions, i.e., standing still, turning, successful movement, wrong movement causing collisions, and particular actions achieving behavioural motivations at a particular time. Figure 3.4 shows these states and their transitions driven by those kinds of motor actions. The No-action state indicates that a virtual creature intends to do nothing at present and hence its body is in a static state. The Turn state means the creature changes its direction by turning. When the body of the creature moves and executes a correct movement, it is in the Movement state. Otherwise, if the body moves but causes a collision, it is in the Collision state. The body arrives at the Target state if its current behavioural motivation is achieved after a particular action. This particular action can be a forward movement that moves the virtual creature into a goal destination, or an action that is the last one executed in a permitted time. The four states of No-action, Turn, Movement and Collision and their transitions compose an action set in practice. When the behavioural motivation is achieved, no matter which action state the body is in, the Target state is entered and the given behaviour has been carried out. All states in the action set can transit to the Target state, and all transits to the Target state are unidirectional and irreversible.

From the state diagram (Figure 3.4), we can see that, for a good performance, the CMC system needs to select an appropriate motor action $a_t = \pi(e_t, s_t, g)$ at each time t so as to obtain the highest probability of reaching the Target state and the least probability of reaching the Collision state, that is,

$$Prob\{s_{t+1} = Target\} \rightarrow 1 \tag{3.3}$$

and

$$Prob\{s_{t+1} = Collision\} \rightarrow 0 \tag{3.4}$$

Therefore, the principal obligation of the CMC system is to employ a rational action selection policy π to each behaviour so as to implement this behaviour

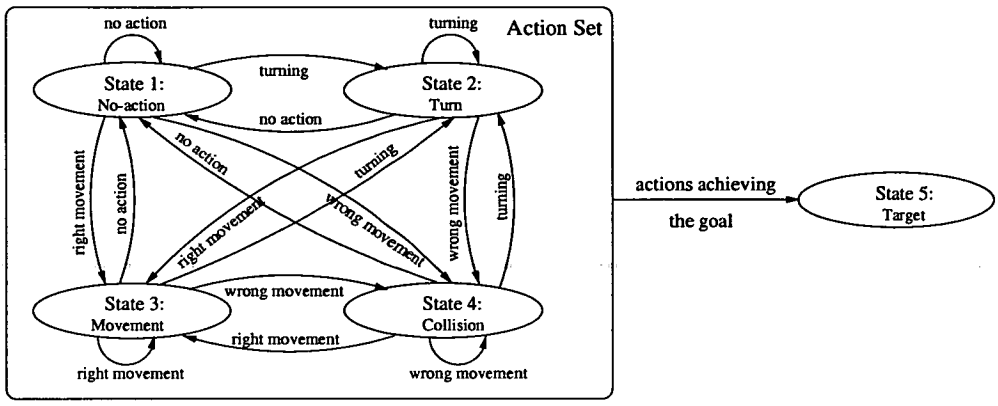


Figure 3.4: State transition diagram for space occupying behaviours

quickly and successfully. At the same time, the function π should comply with some constraints defined by the behaviour, such as the obstacle avoidance constraint (equation 3.4) in space occupying behaviours.

It is worth mentioning that it is possible for the action set of the state transition to have been already launched but for the behavioural motivation to be absent if the Target state is not defined. In this case, the body state transition will loop within the action set endlessly and the Target state will never be reached. This inner cycle in the action set results in restless locomotion in space occupying behaviours. However, even in this endless movement, a unique purpose of obstacle avoidance is still retained, i.e., to satisfy equation 3.4. An example of this kind of movement procedure is endless wandering in space occupying behaviours. In order to terminate endless wandering, a "stop" signal should be initiated from the Motivation module and sent to the CMC system so as to force a direct body state transition to the Target state.

3.5 Potential Strengths

In the last section, we introduced the basic structure and principles of the Computational Behaviour Generation model. The proposed architecture of the CBG model has many potential strengths. For instance, the CBG model can become

adaptive by learning if it exploits its sensory feedback and past experience properly. It can also be multifunctional and perform several functionally related or similar behaviours together. In addition, the CBG model may possess some other strengths, such as autonomy, generalisability, and modularity.

3.5.1 Adaptation

A behaviour control system can possess varied adaptive abilities through several different but complementary approaches (see [102, 103, 65] for a comprehensive review of adaptive behaviours, and the Proceedings of the International Conferences on the Simulation of Adaptive Behaviour for recent results). High-quality sensory mechanisms are one way to enhance adaptation, by gathering useful information for a designated behaviour [37, 39, 86, 119], and by predicting perceptual consequences of a given action in a given sensory context [52]. Flexible selection schemes that choose the behaviour most suitable to the current situation from several possible behaviour reactions are another approach to endow a behaviour control system with adaptation capabilities [20, 27, 92, 161]. Providing adaptive behaviour control in this manner can reflect the changes in motivations [102]. A behaviour control system can have considerable adaptation when it learns from previous action selections that have been shown to be useful or harmful in the past [73, 77, 95, 152, 175]. A behaviour control system would also be able to move on varied terrain and produce various complex postures if it knows how to coordinate and program its motor apparatus appropriately [42, 81, 99]. However, it is obvious that it is not easy to achieve adaptation in these aspects all at once, especially when these aspects are closely related to each other. Indeed, it is not even easy to achieve complete adaptation in just one aspect. In this thesis, we would like to study the adaptive learning ability of decision-making on correct motor action selection in the Strategy module as our first step towards full adaptation in the CBG model.

The main purpose of the Strategy module in the CBG model is to design an appropriate strategy for executing particular behaviours successfully. For different behaviours, the concrete strategies would be different. Nevertheless, generally

speaking, these strategies have the same objective of selecting correct motor actions at every step so as to achieve behavioural motivations quickly and efficiently. This procedure can be simply summarised as a loop between decision-making of action selection and performance evaluation of action execution. This simple strategy, as shown in Figure 3.5, analyses the information input to it first, including perceived environmental information, body states, and commands sent from the higher level, i.e., the Motivation module. The strategy then selects between actions, which are to be elaborated and executed by the lower levels of the CMC system. After action execution, the Strategy module checks to see if the designated behavioural motivation has been achieved successfully. If so, the strategy finishes at a "Success" state. However, in most cases, more than one decision of motor actions would be required for carrying out a particular behaviour. Therefore, the question, "Are the failure conditions satisfied?" (that is, "Are the resources, such as time, energy, etc., running out?") will be asked if the particular behavioural motivation is not yet achieved. If the limited resources are used up, the strategy ends at a "Failure" state, since the behavioural motivation cannot be achieved. Otherwise, the Strategy module makes its next selection of actions toward the behavioural motivation based on the new current input information. This loop continues until the behaviour is carried out successfully or failure constraints are reached.

The above strategy is clearly very simple: the control goes straightforward and does not take any past experience on selecting and executing motor actions into consideration. The Strategy module can become adaptive if it is aware of how to learn from its sensory feedback and past experience. Because the hierarchical architecture of the CBG model is similar to that of the natural behaviour control system, and has mixed information flows of top-down control commands and bottom-up sensory feedback, it has already been implied that the CBG model ought to be able to possess the adaptive learning ability in the same way as the natural system does. In particular, the Strategy module of the CBG should be able to utilise the correct decisions of actions and the mismatch between intended actions and executed results to improve its decision-making ability.

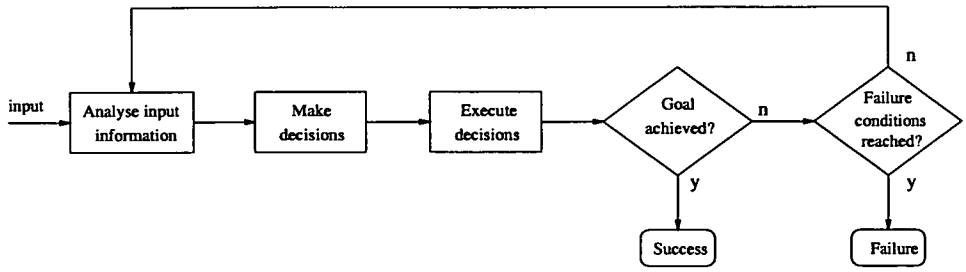


Figure 3.5: A simple strategy

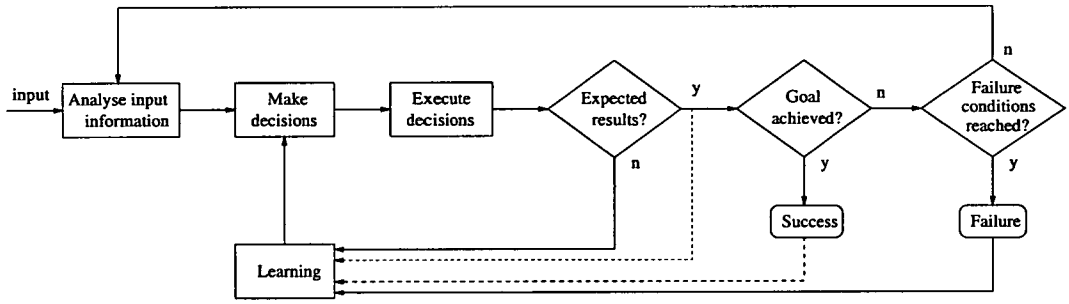


Figure 3.6: An improved strategy with adaptive learning

An improved strategy with additional procedures of adaptive learning is shown in Figure 3.6. This strategy can learn from both successful and unsuccessful experience. The dotted lines in Figure 3.6 illustrate the first kind of adaptive learning, that is, to learn from successful experience. In this case, the Strategy module has made correct decisions of motor action selection, which are just right for a designated behaviour. The knowledge gained about the successful decisions is then kept in memory for future use. This kind of learning is positive, since it derives from the useful information on carrying out a behaviour.

The improved strategy can be even more adaptive if it adjusts its ability through negative feedback from failed execution of an intended action or a behaviour. This kind of learning is marked as solid lines to the learning procedure in Figure 3.6. By comparing intended actions with their actual execution results, the Strategy module can figure out any mismatch between them, and generate a new reasonable policy to correct the errors. If the errors are corrected in a timely

fashion, the improved Strategy can not only save the behaviour control from current difficult situations, but also keep it from the same errors in the future. This kind of immediate adjustment to the strategy during ongoing activity is essential to the execution of the present behaviour.

By taking past experience and sensory feedback into account, the CBG model and the Strategy module hold the potential of adaptive learning to improve generated behaviours continuously. A key point for achieving adaptation by learning is then to design a reasonable learning algorithm that can learn to make suitable decisions of motor actions from both useful and harmful experience. The algorithm we adopt in this thesis is a Multi-agent based Evolutionary artificial Neural network with Lifetime learning (MENL). Detailed information of the work engine of MENL and its adaptive learning ability will be presented in the following chapter.

3.5.2 Multifunctionality

From the Computational Behaviour Generation model described above, we can easily show that it may possess multifunctional ability. With presence and absence of behavioural motivations and the variety of such motivations, the CBG model may be expected to perform a number of different behaviours. The multifunctional ability of the CBG model benefits from shared hardware resources and software knowledge across multiple behaviours.

3.5.2.1 Shared Resources

The hierarchical architecture of the CBG model has provided the material base for the generation of multiple behaviours together. A series of functionally related or similar behaviours can share many hardware resources of the CBG, ranging from sensors and motors, motor invocation procedures (in the Movement module), compound motor programs (in the Program module), and even general strategies (in the Strategy module) for carrying out those behaviours. The hierarchical architecture of the CBG model gives similar behaviours a high probability of resource sharing. When performing a particular behaviour, a general strategy is first generated in the Strategy module. Rather than designing and execut-

ing every action by itself, the Strategy module calls lower levels to carry out its "thoughts". The Program module organises executable motor programs for those "thoughts", and the concrete execution of the programs by motors is initiated in the Movement module. Therefore, once the diagram of strategy generation, motor programming, and motor execution are settled for one behaviour, other similar behaviours that have the same motors, the same motor programs, or the same strategies, can summon corresponding parts of the CBG directly without the need to repeat the exact design work over and over again. To divide the behaviour control into several layered modules according to their functionality makes the multifunctional implementation not only feasible but also economical in both time and resources. In the following chapters, we will see that the generation of several space occupying behaviours shares most resources of the CBG, including sensors, motors, the Movement module, the Program module, and part of the Strategy module. The only difference between these behaviours is in their different action selection policies. The resultant multifunctional implementation of these behaviours is quite efficient and effective.

3.5.2.2 Shared Knowledge

While the CBG model provides the material resources for producing multiple behaviours together, the knowledge shared between these behaviours makes their multifunctional implementation possible. Although shared knowledge is an abstract concept and varies with behaviours of different kinds, we try to explain this concept in this section by using the formal description of the CBG model and examples of space occupying behaviours.

The behaviours in a creature which are generated by a particular CBG can share their generality in many ways. They may have the same motor actions $\mathcal{A}$, the same body states $\mathcal{S}$, or potentially the same environmental information $\mathcal{E}$. Moreover, these behaviours may have much in common in their body states and state transition activities. At least, the intention to choose the correct motor actions at each step so as to reach the Target state quickly is common to every behaviour. Usually there is still much more shared in related behaviours, such as

the common purpose to avoid the Collision state shared in all space occupying behaviours like exploration and goal reaching. For functionally related behaviours, most motor actions may have similar effects on body states. For example, the action set shown in the state transition diagram (Figure 3.4) is common to varied space occupying behaviours. The most obvious difference in multiple functions may lie in their different action functions π, which are caused by different motivations $\mathcal{G}$. The actual decisions on which motor actions to select and execute are dependent on the particular behavioural motivations at that time. Even so, there are still inevitable similarities in the decision-making procedures for related behaviours.

Suppose there are two kinds of space occupying behaviours, $\mathcal{B}1$ and $\mathcal{B}2$, which the CBG model can perform. Both functions have the same motor actions (e.g., to move forward, turn, etc.), the same general knowledge of environments, and the same categories of body states and most of the state transition activities as shown in Figure 3.4. Also, at each movement, both behaviours $\mathcal{B}1$ and $\mathcal{B}2$ are required to execute the best motor actions so as to achieve the Target state quickly and avoid the Collision state simultaneously, that is, to satisfy equations 3.3 and 3.4. However, because these two behaviours have differing motivational signals sent from the Motivation module, the concrete motor actions selected by the CMC system each time may be different, even if all the other conditions of these two behaviours are the same. For example, if the body of the virtual creature sits at place $x1$ in an environment shown in Figure 3.7, it will easily move forward without collisions if its current motivation is only to explore the environment. But if the motivation is to arrive at a destination $G1$ quickly, the body of the creature should turn to the right first and then move forward. The different selection of motor actions is caused by different behavioural objectives. However, if the body is at $x1$ but its goal destination is changed to $G2$, then the CMC system should assert a "Move Forward" signal so as to reach the destination G2 quickly. This command is just the same as that in exploration. In this case, the knowledge about motor action selection (i.e. Move Forward) for two kinds of space occupying behaviours overlaps despite their differing motivations. Another

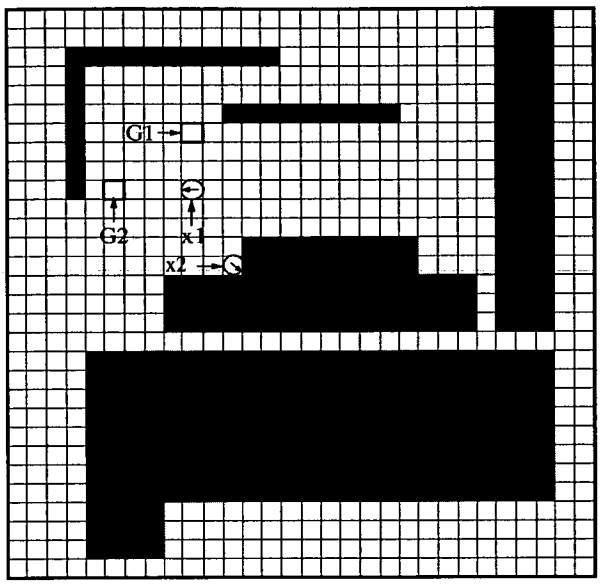


Figure 3.7: Same and different action selection procedures for varied behavioural motivations (see context for the full detail).

example of knowledge sharing between those two space occupying behaviours is when the body sits at a place $x2$ and faces to the southeast. Here, if the body moves forward, it will definitely crash into the obstacle. In this emergent situation, obstacle avoidance takes precedence over accomplishing any other task. So, whatever the behavioural motivation is at that moment, the CMC system should initiate a "Turn" command first to force a release of the body from the predicament. This kind of emergent skill of obstacle avoidance is fundamental to all space occupying behaviours, and is in fact shared by all of them.

In addition to the above examples, there is much other knowledge shared in various kinds of space occupying behaviours. Due to the resources and knowledge shared by functionally related behaviours, the CBG model ought to be able to perform these behaviours easily and efficiently if what it learns for one behaviour can be used for others. Similarly to potential adaptation, the concrete implementation of multifunctionality in the CBG requires a sensible algorithm to select appropriate actions for varied behaviours. In Chapter 5, we will present our techniques and experiments on using the MENL learning algorithm to learn

suitable decision-making policies for multiple space occupying behaviours in the same CBG model.

3.5.3 Other Strengths

In addition to adaptation and multifunctionality, the CBG model has some other potential strengths, such as autonomy, generalisability, and modularity.

A certain autonomy is brought directly by the adaptive and multifunctional abilities of the CBG. After the CBG has learned how to use its past experience and sensory feedback to make correct selections of motor actions, the CBG can behave in various situations without human interference. Similarly, after the CBG has learned how to draw inferences about other behaviours from an already learned behaviour, it can perform many related behaviours based on its own pre-learned knowledge, so the human designer won't have to design repeated details for every behaviour. Of course, autonomous decision-making as to the correct action at any one time is only one type of autonomy, the tip of the "autonomy" iceberg. In order to achieve complete autonomy, the CBG needs to introduce autonomy into every other part, such as sensing, motivation generation, and motor programming and execution.

Due to its hierarchical architecture, the CBG has also the strength of generalisability. The multifunctional ability of the CBG is one kind of generalisability: the CBG can be generalised to more than one behaviour by taking advantage of the shared knowledge and resources across these behaviours. The generalisability of the CBG is also present in its easy extension to a wide range of virtual creatures. Although the sensory and motor apparatus may be varied from one virtual creature to another, the general strategies and/or the motor program may remain the same, since it is the Movement module that finally invokes specific motors to perform. The Movement module performs as an abstraction between the higher level for planning and the lower level for execution. Therefore, some parts of behaviours of the CBG model can be executed in a creature-independent way. The generalisability of the CBG to both various behaviours and virtual creatures would save the human designer from laborious design work and enrich

the variety of virtual creatures and behaviours to be implemented.

Another advantage brought by the CBG's hierarchical architecture is its modularity. The modularity of the CBG is obvious since components of the architecture can be replaced with others in a relatively straightforward manner. For instance, the action selection strategy used in the Strategy module can be directly changed from hand-crafted rules to autonomous learning algorithms. As long as these strategies produce the same functionality, the rest of the CBG won't be affected at all. Similar replacement can also happen in other modules, such as Motivation, Program, Movement, and even Sensor and Motor systems. The modularity of the CBG is particularly useful in extending the CBG's generality to different behaviours and virtual creatures.

The above discussions in Section 3.5.1 to 3.5.3 have shown that the CBG model can possess many potential strengths, including adaptation, multifunctionality, autonomy, generalisability and modularity, provided there are appropriate approaches to implement them. In this thesis, we concentrate on the implementation of adaptation and multifunctionality of the CBG by learning. Research on the other potentials of the CBG is our future work.

3.6 Space Occupying Behaviours

To illustrate how the CBG model works, we will give an instantiation of the CBG model when it is used to generate space occupying behaviours in the following section. Before presenting the instantiation, however, we briefly introduce what space occupying behaviours are and why we choose them as sample behaviours for the CBG model and the following MENL learning algorithm to learn to implement.

In nature, animals spend much of their time moving from here to there in order to reach their shelters or some other destinations (i.e., navigation or goal reaching), explore surroundings (i.e., exploration), forage for food sources (i.e., foraging), or sometimes just move around randomly (i.e., wandering). The be-

haviours are the process of determining and maintaining a course or trajectory from one place to another, and hence called *space occupying behaviours* in this thesis. Space occupying behaviours are functions fundamental to natural animals. The survival of a natural animal is contingent upon the adaptive skills of quick and safe space occupying movement.

Space occupying behaviours are also fundamental to artificial creatures. They have received much attention in both the robotics and virtual creature literature. (See [159] for a review on the recent work on simulating space occupying behaviours.) Space occupying behaviours have been well studied with the assistance of an environmental map or landmarks in the environment. An environmental map provides an explicit global representation of the environment, and based on it, routes or trajectories of movement can be planned. An environmental map can be built in a previous phase or from geometrical information gathered as the robot or virtual creature travels [8, 116, 118, 158, 160, 164]. A variety of methodologies have been proposed in this context, but potential problems still remain. Besides the prior-knowledge required in creating maps, the effort involved in revising a map when any change occurs in the environment can be substantial. Moreover, this approach does not correspond to reality where natural creatures do not always have a map of their environment when carrying out space occupying behaviours. Some approaches used landmarks to obtain a local representation of the environment [14, 87, 96, 114]. This representation comprises a topological modelling of the environment. Usually a robot or virtual creature acquires a graphical representation of landmark types as it moves in the environment. The landmark scheme is likely to improve robustness in space occupying behaviours, but the behaviours can only be executed in the environments that have landmarks and when the landmarks are discernible.

In addition to the behaviours based on landmarks or a map of the environment, space occupying behaviours can also be generated without specific environmental knowledge. In nature, living creatures can move promptly and appropriately by utilising the environmental information they presently perceive. For instance, vision is a popular means of collecting necessary information from the environment.

Sometimes only the current visual information is enough for living creatures to decide what to do next (e.g., finding available places and avoiding obstacles). When the environment is new, or the goal location is out of sensory field, many living creatures including bees, birds, fish, turtles, etc., can determine their move direction or distance or even their global position by using some other tools to assist, such as a sun compass or magnetic sense [61, 89, 165]. The creatures achieve their goals through tentative movements and their movement skills are improved after they obtain more and more experience. Such space occupying behaviours may be less efficient or optimal compared with planned ones, but they are robust and general to a wide range of situations. These skills are especially useful when the creatures are out of the niches that they are familiar with or when the environment is completely unknown. These space occupying behaviours are complementary and coexist with map-based and landmark-based behaviours in many different animal species [126]. The space occupying behaviours, which are based on local environmental information and general to various situations, are called *general space occupying behaviours* in this thesis.

The CBG model combined with the MENL learning algorithm has been trained to learn general space occupying behaviours as a test of its adaptive and multifunctional behaviour generation. In addition to the fact that general space occupying behaviours are important skills to many animals, there were a number of reasons to choose them as model behaviours for the CBG and MENL to learn:

- General space occupying behaviours are adaptive

  General space occupying behaviours are adjustable to different environments and situations. They can be adjusted to new circumstances and continuously improved through practice [147]. This adaptation of general space occupying behaviours is suitable to test whether the CBG combined with MENL is able to learn behaviours adaptively.

- General space occupying behaviours are full of variety.

  General space occupying behaviours are a general designation of behaviours

that involve movement from one place to another according to current environmental information. These behaviours include exploration, foraging, goal reaching, wandering, and some others, which are differentiated from their different behavioural objectives. The achievement of these behaviours by an animal involves the same motors and motor actions but different decisions about which motor action to execute at a particular time. The commonalities and differentiations between these functionally related behaviours are good examples to test the multifunctionality of the CBG model and MENL.

- General space occupying behaviours are robust

  As explained above, general space occupying behaviours are not specific to any environment or object. They are general and robust to a wide range of situations. Therefore, virtual creatures that have general space occupying ability should be able to decide for themselves how to achieve their behavioural goals in various environments without help from a human designer. Such robust behaviours can enhance virtual creatures' survival ability in dynamically changing environments.

- General space occupying behaviours are of assistance to many other behaviours.

  Although general space occupying behaviours differ somewhat from other space occupying behaviours, such as those based on landmarks or a map of the environment, the implementation of general space occupying behaviours can assist the achievement of other behaviours. For example, when full knowledge of the environment is absent, general exploration can collect the necessary place information for the construction of a map. When a landmark is too far to sense, general navigation can help the creature to reach the areas that have landmarks. This kind of self-sufficiency can help virtual creatures to accomplish complex behaviours in their environments.

3.7 An Instantiation of the CBG Model for Space Occupying Behaviours

In this section, we present an instantiation of the CBG model where it is used to produce some general space occupying behaviours, such as exploration, goal reaching, and wandering. The implementation of the space occupying behaviours is decomposed into a sequence of procedures from general to specific for the CMC system to execute. Meanwhile, the behaviours are implemented with assistance of all the other parts of the CBG model. This instantiation should give us a clear image of how the CBG model can be implemented to generate concrete behaviours for virtual creatures. The virtual creature adopted here is an abstract two-dimensional bot with an arrow pointing in the direction in which it is facing, as shown in Figure 3.8. When the CBG model is applied to the virtual creature to generate space occupying behaviours, the implementation details of every part of the CBG model are summarised as follows.

3.7.1 Synthetic Visual Input Pattern

The Sensor system of the CBG model can be constructed with different kinds of sensors. In this thesis, we are mainly interested in simulated vision sensing.

Simulated vision sensing is a direct and natural method to acquire environmental information. This is mainly because vision is a rich source of information concerning a creature's living world; and it can be exploited in a very wide range of behaviours. In the study of virtual creatures, *synthetic vision* is the one often used [20, 116, 132, 133, 162]. Because the computer graphics system contains all the information about the simulated environments and objects for their rendering and display, synthetic vision can directly use this information from the graphical system. For instance, for each pixel in the rendered scene viewed by a creature, synthetic vision can easily conclude its position, the distance between the eye and the point from which it is projected, and the object which it belongs to. However, it should be noticed that not all information can be provided to the virtual lives directly. It is necessary to simulate not only the abilities but also the

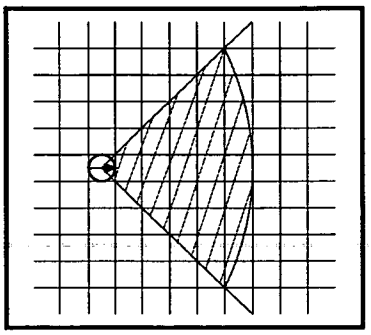


Figure 3.8: A virtual creature and its visual input. The circle represents the virtual creature and the arrow its view direction. The shaded area is the visual field of the creature.

limitations of the perception systems of animals for producing natural behaviours [153]. Therefore, sensible choices should be made on what sensory information can be provided to the virtual creatures and what information cannot.

In the work reported here, the virtual creature is equipped with a straightforward visual sensor. Its field of vision ranges over 90 degrees and five squares in distance, as shown in Figure 3.8. This visual sensor simulates the current visual field of the creature and is concerned with only one kind of information, which identifies whether a square is occupied by an obstacle or not. This is mainly because we believe a primitive movement can be made on the information about the availability of places in the visual field, without knowing some other specific knowledge such as obstacle shape or size. Therefore, each point in the simulated visual field of the virtual creature has only one of two values: 0 represents a place occupied by obstacles, and 1 a free place. This information can be easily obtained from the graphical system. A visual input value $\mathcal{V}$, which is a real number normalised in the interval [0,1], then results from a weighted sum of all the points in visual field:

$$\mathcal{V} = \sum_{x}(2^{-d(x)}a(x)) \quad a(x) = \{0,1\}, \ \forall x \ in \ the \ visual \ field \tag{3.5}$$

where $d(x)$ is the distance of a point x to the current position of the virtual

creature, and $a(x)$ indicates the availability of the point x.

The points in the visual field have gradually decreasing weights from near to far with respect to the creature. Therefore, close areas in the visual field will have more significance than far areas. The result of combining all the places in the visual field gives an input value $\mathcal{V}$ to feed the CBG.

Because the visual sense adopted in the CBG model is related to the availability of places in the visual field, it is independent of specific environments and objects. It is therefore suitable to any situations.

3.7.2 Virtual Motors

At present, the virtual creature has two kinds of virtual motor for executing space occupying behaviours: *Motor_Move* and *Motor_Direction*. The Motor_Move motor is used to move the creature straight forward for one square. The Motor_Direction motor takes one parameter, α, and turns the creature's body clockwise to a particular angle α. When the CMC system in the CBG model sends a motion command to a virtual motor, this motor makes the corresponding action on the graphical model of the creature. Specifically, it is a reposition and reshaping of the virtual creature in the graphics model.

3.7.3 The State Module

According to the body state classification introduced in Section 3.4.3, the creature has five major state categories in space occupying movement: No-action, Turn, Movement, Collision and Target. The State module records not only the current body state and its transitions, but also past state activities. The latter are used to store the related information about body states and motor actions the creature has made when performing a particular behaviour. It serves as the "working memory", a temporary storing of information, to guide a future action. For example, when the CBG executes space occupying behaviours, the State module will record the position of the accessed place, the body state of the creature, and the result of executing the chosen motor action at each time step. Though

information like sensory input for places accessed in the past may be useful for some other behaviours, it is not necessary for space occupying behaviours. The useful information saved in the State module can be accessed by every part of the CMC system and the Motivation module for them to decide what to do at the next step.

3.7.4 The Motivation Module

The Motivation module is responsible for establishing behaviour goals for the virtual creature. In this thesis, it acts primarily as an interface to a human commander. When the human commander informs the Motivation module of a behavioural motivation, the Motivation module generates and sends a corresponding motivational signal to the Computational Motor Control (CMC) system to achieve this motivation. In the following experiments conducted in Chapter 4 and 5, the Motivation module will be informed of three candidate values (wandering, exploration and goal reaching) and their various combinations. This result in three corresponding space occupying behaviours executed both individually and jointly.

3.7.5 The Computational Motor Control System

3.7.5.1 The Strategy Module

The top level of the Computational Motor Control (CMC) system, the Strategy module, analyses the motivational signal sent from the Motivation module and constructs an overall strategy and motor plan for achieving the corresponding motivation. The strategy adopted in the Strategy module is the adaptive strategy explained in Section 3.5.1 and shown in Figure 3.6.

Motor Action Selection

For different behavioural motivations, the policies of choosing suitable motor actions are usually different. Thus, the module of "Make decisions" in Figure 3.6 will be implemented differently. For example, when selecting actions for

carrying out the goal reaching behaviour, the Strategy should consider both the possible reactions (responses) from the environment and body states, and the distance of the place to be accessed from the goal destination. When executing exploration, the responses from the environment and body states are the same, but the information about whether a place to be accessed has been visited before is another factor to consider. The Strategy needs to choose an action that will access an unexplored place in the environment if there is any. For the behaviour of wandering, the action selection is relatively easy. Because there is no obvious behavioural objective in wandering, only the responses from the environment and body states will be considered. The different factors involved in motor action selection result in different actions to be chosen for achieving different behavioural motivations.

Despite the different motor action selection, the remaining parts of the adaptive strategy used for carrying out varied space occupying behaviours are quite similar. This similarity again shows that functionally related behaviours have some knowledge in common relating to their execution.

Motor Actions

The virtual creature adopted in our work is assumed to make at most one step at each time interval in its movement. Accordingly, there are eleven motor actions designed in the CBG model. These actions are *"move one square straight forward"*, *"move diagonally one square to the right forward"*, *"move diagonally one square to the left forward"*, *"remain stationary"*, and turn to the other seven different directions. A pictorial explanation of these actions is shown in Figure 3.9.

Of eleven motor actions, "move diagonally one square to the right forward" and "move diagonally one square to the left forward" are compound movements, which need to call more than one motor in their execution. For executing those compound movements, the creature should first turn to the correct direction by using the Motor_Direction motor and then move one step forward by using the Motor_Move motor.

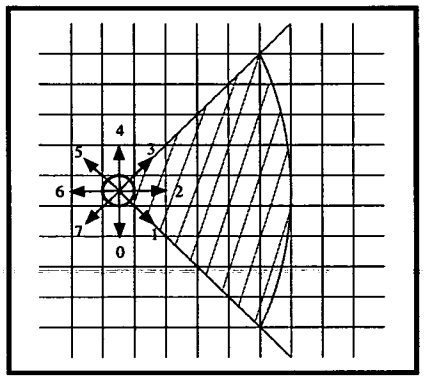


Figure 3.9: Eleven motor actions of the virtual creature. The creature can move one step forward in its visual field following directions 0, 1 and 7 respectively, and turn to the other 7 directions other than its current direction 0 by turning around. The last action choice is to remain at its original place.

Compound motor actions are very popular in natural animals. Animals can decide between numerous sophisticated movements and the implementation of these movements may require the use of many individual "motors" many times. To choose compound movements is obviously quicker and more efficient than to choose single motor movements one by one. In the CBG model, because the decision-making of motor actions is relatively independent of the concrete execution of these motor actions, the Strategy module is able to decide between compound movements and leave the implementation details of these movements to the following modules of the CMC system.

3.7.5.2 The Program Module

The selection of motor actions made by the Strategy module is first sent to the Program module for further elaboration. The Program module is responsible for designing detailed motor programs in space and time for the abstract selection of motor actions made in the Strategy module. The Program module decomposes those decisions into detailed programs that can drive the motors. In the eleven motor actions, the decision to move one square straight forward can be achieved by calling the moving motor (Motor_Move), and the decisions to turn to the

other seven directions can be achieved with a direct invocation of the direction changing motor (Motor_Direction). The movement pace of Motor_Move is one abstract unit of movement each time. The turning scale of Motor_Direction is a value between 1 and 7, which is an abstract description of the direction to be reached, according to the direction labelling shown in Figure 3.9.

The Program module needs to divide the decisions "move diagonally one square to the right forward" and "move diagonally one square to the left forward" into sequential movements of direction change and straight forward movement, so the motors of Motor_Direction and Motor_Move will be invoked in turn. The turning values of Motor_Direction for those two compound decisions are set to 1 and 7 respectively. The movement paces of the Motor_Move motor are still one abstract unit of movement for both compound decisions.

The motor programs and their coefficients (e.g., the movement pace and turning values) generated by the Program module are still relatively abstract and separated from the real execution of motors. Those motor programs are then sent to the Movement module for a concrete execution of the virtual motors.

3.7.5.3 The Movement Module

The concrete invocation of the creature's motors is implemented in the Movement module by setting real turning angles and movement distances. The Movement module interprets the abstract coefficients sent from the Program module into actual turning and movement parameters acting on the graphical model of the creature. The movement distance of the Motor_Move motor is actually one grid distance of the virtual environment. The turning values ($1 \sim 7$) of the Motor_Direction motor are translated into $45 \sim 315$ clockwise degrees. After receiving invocation signals and real graphical parameters from the Movement module, the Motor system launches the corresponding virtual motors to reposition and reshape the graphical elements of the virtual creature. In consequence a sequence of movements can be generated for carrying out various kinds of space occupying behaviours.

The translation of abstract motor actions into real graphical parameters makes

the decision-making of motor action selection and their decomposition to be separate from the actual graphical model of the virtual creature. As a result, the CBG model can be applied to different kinds of virtual creatures and motors without much redesign.

In this section, we explained how to set the CBG model concretely so as to produce particular behaviours such as space occupying behaviours. Through this explanation, we can see that the concrete execution of a behaviour is achieved by the coherent collaboration of every part of the CBG model. An important part of which motor actions to choose at a certain time for a particular behavioural motivation is not presented here. This part is accomplished by the MENL learning algorithm and its implementation will be introduced in the next chapter.

3.8 Conclusion

The Computational Behaviour Generation (CBG) model proposed in this chapter provides a general and complete framework of behaviour control for virtual creatures. This model has taken its inspiration from and has been designed based on the natural behaviour control system in the brain. In particular, the CBG model utilises a hierarchical Computational Motor Control (CMC) system to perform the whole procedure of selection, programming and execution of motor actions. This hierarchy provides a top-down control scheme in which higher levels produce general commands for lower levels to implement without detailed description of concrete implementation of lower level elements. The CBG model designed in this manner has clear generality in its behaviour control and can therefore be applied to a wide range of virtual creatures. In addition to the top-down control, a bottom-up feedback is formed in the CBG model, as the sensory system continuously reports to the CBG the changes in the external environment and internal body states after every motor action execution. By means of the top-down and bottom-up information flows, the CBG model can compare its intended actions with actual movements and learn from its successful or unsuccessful experiences.

In consequence the behaviours generated by the CBG model can be adaptive by learning. Due to its hierarchical structure, the CBG model supplies a great amount of commonality in functionally related or similar behaviours, including both hardware resources and software knowledge. The CBG model may therefore be multifunctional, if it takes what it learns from one behaviour to another. In addition to adaptation and multifunctionality, the CBG model also possesses some other potential strengths, such as autonomy, generality, and modularity.

In the next chapter we will present the learning algorithm of Multi-agent based Evolutionary artificial Neural network with Lifetime learning and introduce how this algorithm can be used in the CBG model to produce adaptive behaviours.

Chapter 4

Learning Single Behaviours: Combining the CBG Model with the MENL Learning Algorithm

In the previous chapter, we explained that the Computational Behaviour Generation (CBG) model can become adaptive, provided it has a sensible learning algorithm that can learn to choose suitable motor actions for generating behaviours. The learning algorithm adopted in this thesis is MENL, a Multi-agent based Evolutionary artificial Neural network with Lifetime learning. This chapter gives a full explanation of the learning algorithm MENL and its adaptive learning of single behaviours when it is combined with the CBG model. (The learning of multiple behaviours by the CBG combined with MENL will be presented in the next chapter.) In particular, the next section is a review of the recent work on evolved behaviours and reinforcement learning, which are related to the adaptive learning of MENL. Section 4.2 outlines the main ideas behind the multi-agent based evolutionary artificial neural network with lifetime learning. Section 4.3 describes the implementation details of MENL when it is employed in the Strategy module of the CBG model to learn correct motor action selection for generating space occupying behaviours. In this chapter, the CBG combined with MENL is required to learn exploration behaviour independently, that is, to

access places in an unknown environment as far as possible. The experimental work is presented in Section 4.4. The experimental results have shown that the CBG with MENL has successfully learned exploration in various unknown environments. Some efficient and believable behaviours emergent in the experiments are introduced in Section 4.5. The last section concludes this chapter.

4.1 Related Work

Evolved behaviours and reinforcement learning are two important approaches to generating adaptive behaviours for artificial animats. Our MENL learning algorithm is associated with both of them because it uses not only evolution strategies but also sensory feedback to shape behaviours. Based on a skillful combination of evolutionary learning and reinforcement learning, MENL also utilises the agent technology and the lifetime learning in its design so as to achieve improved adaptation in behaviour generation. In this section, we review the related work on both evolved behaviours and reinforcement learning and place our work in the context. Instead of an exhaustive survey of a very large domain, we would like to focus on those works closely related to our study.

4.1.1 Evolved Behaviours

The idea that an animal's behaviour is partially determined by its genome and hence evolvable through natural selection has inspired much research work on using evolutionary procedures to develop the mapping between environmental situations and actions [16, 38, 45, 84, 115, 105, 134]. The large number of evolved behaviours includes obstacle-avoidance, locomotion, wall-following, box pushing, finding food sources, etc (see [57, 104] for a comprehensive review). The rich variety of structures that have been put under evolution include directed graphs, Lisp code and artificial neural networks. Directed graphs have been used to represent the morphology and behaviours of virtual creatures, and evolved to achieve specific behaviours such as swimming, walking, and jumping [148, 149]. The Lisp code, also called *genetic programming*, has been applied to recreate

the patterns of locomotion of a lizard [84], determine the optimal number and orientation of the sensors for developing corridor-following behaviours [134], or design both controllers and morphology of robots [80, 88, 91].

In order to avoid the artificial interference brought by the human designer, it is believed that the primitives manipulated by the evolutionary process should be at the lowest level [69]. Therefore, artificial neural networks composed of the very basic units of neurons are frequently chosen as the building blocks for evolutionary learning of behaviours. The genotype-phenotype mapping of evolutionary algorithms can be consequently implemented in a classical neural network [51, 50, 120, 173] or in a dynamic neural network [38, 63, 68, 178]. Cliff, Husbands and Harvey, for instance, employed an extended genetic algorithm with variable-length genotypes to evolve the number of hidden units and specific connections of recurrent real-time artificial neural networks for controlling robots to avoid obstacles or to reach a particular target [68, 69]. One of their major claims was that artificial evolution of neural networks represented a better choice for the development of autonomous behaviours than design by hand [38]. Cliff also provided a theoretical background for the study of simulated organisms in a closed environment and presented a concept of *Computational Neuroethology* as an attempt to relate behaviours with the activities of artificial neural mechanisms [36]. In this study, Cliff concluded that connectionist models could only be meaningful if they are embedded in a sensorimotor system.

Floreano [51] studied the evolution of a feedforward neural network which exhibited a nest-based foraging behaviour. Since the fitness function was simply the number of food objects eaten, the location of the nest and the ability to periodically visit it were indirect achievements. A similar result was obtained in the experiment of Floreano and Mondada [50], in which the periodical return to a battery recharge station was indirectly achieved in the evolutionary learning of a discrete-time recurrent neural network. The experimental results showed that more complex behaviour emerged by reducing the constraints imposed by the fitness function.

The work by Nolfi also concluded that the lesser power of current artificial

evolution models might be due to the lack of some important properties of natural evolution, and the usage of a very specific selection criterion implied that the evolutionary process was used in at least a partially distorted way [115]. In his research work on a garbage collecting robot that was trained to keep an arena clean, Nolfi carefully "canalised" the evolutionary process in the "right" direction, by favouring the emergence of the required competencies. Through experiments, Nolfi also found that the amount of canalisation pressure should be kept as small as possible.

Inspired by biomimetic processes such as cell division, axonal growth, and protein synthesis regulation, evolutionary algorithms have also been used to evolve developmental procedures of neural controllers [46, 63, 105, 106]. Michel, for instance, applied genetic algorithms to evolve morphogenesis production rules which were decoded into a dynamical neural network driving a mobile robot [106]. Eggenberger designed an artificial genome composed of regulatory units and structural genes [46]. The activities of structural genes were regulated by regulatory units in evolution and resulted in a network of cells through cell differentiation, cell construction, cell division and cell connection. The developed network were successfully linked to the sensor and motors of a real robot. A grammar tree based cellular encoding with syntactic constraints was proposed by Gruau et al. [62, 63]. Based on this encoding, a neural network was finally generated from a single cell via various kinds of cell divisions. During the evolution or development of cells, the fitness of evolutionary individuals were given by the experimenter interactively by hand. The AnimatLab at Paris has proposed an evolutionary paradigm of SGOCE [105], which is an integration of an axonal growth process [32, 163] and the cellular encoding of Gruau et al. A series of work has been conducted in this lab to use SGOCE to evolve neural controllers for generating behaviours such as rolling [35], walking [47, 48], swimming [76] and flying [44]. Some of the behaviours were produced in an incremental way by taking advantage of the geometrical nature of the developmental controllers. In [82, 83], for example, locomotion controllers were first generated by evolution. Neural controllers for gradient-following and obstacle-avoiding were subsequently evolved,

whose neurons established flexible connections with neurons of the evolved locomotion controllers. Experimental results have shown that the controllers obtained from the first stage were less efficient than those obtained from the subsequent stage, and meanwhile, the evolution of evolving all the controllers simultaneously had worse average performance or more evolution time than the incremental evolution [35, 44]. By means of incremental evolution, neural controllers can become increasingly complicated and hence control more and more complex behaviours.

While most research on evolutionary robots has employed genetic algorithms (GAs) or their variations, Salomon studied the application of evolution strategies (ESs) to the evolution and optimisation of different controllers for Braitenberg vehicles [137]. The aim of the application was to make the vehicles move forward quickly with obstacle avoidance in an arena. Compared with other research on Braitenberg controllers that applied GAs, the experimental results have shown that the ES-based approach was much faster and more competent than GA-based approaches, especially when encountering epistasis problems.

Our approach of MENL learning is much influenced by the above research work on evolved behaviours. In particular, a feedforward neural network is evolved in MENL via evolution strategies to learn appropriate situation-action mapping for the generation of behaviours. The objective of the behaviour to be achieved is a factor in the fitness function guiding the evolution of the neural network. Different from evolving behaviours in a traditional way, the MENL learning algorithm also takes into account the sensory feedback on executed actions (including environmental and body state feedback) in its action selection learning. The sensory feedback is used as another important factor to discover general properties of behaviours of the same kind. The learned behaviour is therefore also shaped by the immediate feedback from the environment and body states. In this sense, our approach is also related to reinforcement learning.

4.1.2 Reinforcement Learning

The main purpose of reinforcement learning is to learn an action policy, or an associative mapping from situations to actions by maximising a scalar reinforce-

ment of the task performance from environments. There are two main strategies for solving reinforcement-learning problems: value function methods and policy space methods (see [78, 112] for a comprehensive review).

The value function methods attempt to learn the optimal policy via a value function, instead of a representation of the policy, which returns the expected cumulative reward for the policy from any state. Q-learning, a technique for propagating rewards temporally across sequences of actions [170, 171], is one of the most well known value function methods. Much of the work on robot learning is derived from it. Mahadevan and Connell used Q-learning with some clustering techniques to train a mobile robot to push large boxes for extended periods of time [95]. The clustering techniques were weighted Hamming Distance and statistical clustering that generalised rewards spatially across similar states. Mataric described a robotics experiment with a high dimensional state space based on Q-learning [97]. Four mobile robots travelled within an enclosure to collect small disks and transported them to a destination region. Pre-programmed signals called *progress estimators* were used to break the monolithic task into subtasks. State space was also quantised into a small number of discrete states according to pre-defined boolean features of the underlying sensor. The performance of the Q-learning policies was almost as good as a simple hand-crafted controller for the job. In order to reuse the same sequence of situation-action pairs so as to back-propagate delayed rewards, Sutton studied a class of reinforcement learning architectures called DYNA, which included an internal world model in learning [151, 152]. DYNA simultaneously used experience to modify the world model by *relaxation planning* and the value function by *temporal difference reinforcement learning* [150], and used the world model to adjust the value function. When DYNA was used to navigate in a maze, it required fewer steps of experience than Q-learning to arrive at an optimal policy, but more computational effort at the same time.

Reinforcement learning can also be achieved via policy-space methods, which maintain explicit representations of policies and modify them through a variety of search operators, such as evolutionary algorithms. Grefenstette et al. pre-

sented the SAMUEL system that combined evolution with aspects of temporal difference reinforcement learning [59]. They used a rule-based single chromosome to represent a policy, i.e., a situation-action mapping. So, each individual of the evolutionary population was a policy represented as a rule set and each gene was a rule that maps the states of the world to actions to be performed. SAMUEL has been applied to learn a series of behaviours for robots and other autonomous vehicles, including collision avoidance and local navigation behaviours [138, 139, 140], and herding a second robot to a "pasture" [140]. Several other systems used *Classifier Systems* to find food, avoid obstacles, or produce goal-seeking sequences [23, 70, 175]. In classifier systems, a policy is represented by a set of distributed if-then rules called *classifiers*. Each chromosome represents a single decision rule that maps part of the sensory input to an appropriate action and the entire population represents the agent's policy. Every classifier has a statistic called *strength* that estimates the utility of the rule. Genetic Algorithms are usually applied to highly fit classifiers to generate new rules.

In order to obtain higher generalisation over the input space, some work used neural networks to approximate a situation-action policy [15, 174, 177]. A neural network for a decision policy is represented as a sequence of real-valued connection weights, and these weights are continuously optimised via an evolutionary algorithm to search out an optimal policy. This representation requires little modification of the standard evolutionary algorithms. In the SANE system [110, 111], two separate populations were maintained and evolved: a population of neurons and a population of network blueprints. The evolution of neurons provided evaluation and recombination of the individual neurons which were used to construct dynamic neural networks. The evolution of network blueprints then searched for effective network combinations by these neurons. SANE explicitly decomposed the neural network search problem into several parallel searches for effective single neurons. The SANE system has been shown to be effective in game-tree search [109] and obstacle avoidance learning in a robot arm [110]. In evolutionary algorithms for reinforcement learning, almost all the fitness functions reflected accumulated rewards received during the course of interaction with the environment.

Fitness might also reflect effort expended, or amount of delay.

Similar to reinforcement learning, our approach of MENL learns behaviours by maximising sensory feedback from environments. MENL is especially similar to the evolutionary algorithms for reinforcement learning because it also uses evolutionary algorithms to learn an implicit representation of the situation-action policy. However, MENL differs from the traditional reinforcement learning in several fundamental ways. The rewards that MENL obtains are simple and general responses of not only environments but also body states to executed actions. The responses are not specific to a particular behaviour. Instead, they are general information common to behaviours of the same kind. For example, when learning space occupying behaviours, the sensory feedback only indicates that successful walk is better than direction change and direction change is better than no moving and collision making. This information suits all space occupying behaviours like exploration and goal reaching, and more importantly, it is easy to design and obtain. This is in contrast to the difficult reward calculation and credit assignment in reinforcement learning [94]. The fitness function adopted in MENL is also different from that in traditional reinforcement learning. The fitness function of MENL consists of both sensory feedback and behaviour objectives. While the sensory feedback provides the generic information of behaviours, the objective of the behaviour to be learned helps the creature to learn how to perform a particular behaviour through evolutionary learning. In consequence, the MENL learning algorithm can be used to learn more than one behaviour easily and efficiently, by changing the behaviour objective in the fitness function. (The following chapter will support this claim.) The characteristic of multifunctional learning in MENL is in contrast to reinforcement learning that has difficulties to deal with varying goals [94]. If the goals change, almost everything of reinforcement learning has to be reset. The inflexibility in reinforcement learning is not suitable to virtual creature applications that may have various goals and behaviours to achieve.

In order to improve the adaptation of a behaviour to be learned, and especially to improve the robustness and generalisation of the behaviour, the MENL learn-

ing algorithm has also adopted multiagents in its evolutionary learning and this learning is kept through the lifetime of an animat. The learned behaviours therefore can be continuously improved and suited to a wide range of situations, as demonstrated by the following experiments.

In the next section, we will introduce the basic design of MENL in detail.

4.2 Multi-agent Based Evolutionary Artificial Neural Network with Lifetime Learning (MENL)

Multi-agent based Evolutionary artificial Neural network with Lifetime learning (MENL) is an evolutionary artificial neural network that can continuously learn how to select suitable motor actions for generating designated behaviours based on its own knowledge and experience. This learning algorithm has a strong ability to adapt to continuous changes in the environment. To enable this ability, many technologies and concepts have been introduced into MENL. Particularly, Evolution Strategies (ESs) are used to evolve the parameters of the artificial neural network. The fitness function of MENL is a relaxed and general design composed of both behaviour objectives and sensory feedback on executed motor actions. Therefore, the learning of MENL is guided by not only the objective of a behaviour but also the environment that provides the feedback. In order to take advantage of the whole population information, a batch of agents is maintained in evolutionary learning, each of which is an evolutionary neural network individual. Through constant interaction with the environment, these agents cooperate and compete with each other for correct decision-making of motor action selection. These agents are subject to evolution via evolution strategies in the absence of a correct decision, and this kind of evolutionary learning continues through the lifetime of the ANN. In consequence, MENL obtains a more enhanced and general ability as the evolutionary learning proceeds during its lifetime.

In this section, we introduce the strategies used in MENL, including artificial neural networks, evolutionary learning, multiagents, lifetime learning, and a relaxed and general fitness function design. When introducing these strategies, we

also explain the reason we use them and the ways to implement them in MENL. The entire work engine of MENL is then presented. At the end of this section, we briefly explain how adaptive learning is achieved when MENL is combined with the CBG model.

4.2.1 Strategies Used in MENL

4.2.1.1 Artificial Neural Networks

In animals that have neurons, neural networks are always involved in behaviours [147]. These networks at least act as the conductor that sets off effector responses. Usually they are responsible for the transference and transformation of environmental information received by the receptor to action commands for the effector to carry out. The pathway and connection in a neural network (receptors-neurons-effectors) determine what the action responses will be in the end. Neural networks can be modified. By establishing alternative routes of information flow, they can revise old responses and produce new responses so as to adapt to a new situation. At the same time, the old responses can be recalled when they are required. Neural networks are the cornerstone of behaviours, especially in higher invertebrates or vertebrates.

By simulating some of the structures and characteristics of biological neural networks (BNNs), artificial neural networks (ANNs) in computer science possess some functions of BNNs on certain levels. ANNs can also function as an internal engine between simulated receptor and effector for producing animal-like behaviours. Due to their outstanding ability for information abstraction, ANNs are able to learn internal relationships between environmental information and appropriate action responses. Knowledge of relationships is then implicitly stored and distributed in neurons and connection weights. When encountering new situations, ANNs can also revise old knowledge and fuse new information so as to deal with these situations competently, by adjusting the structure and parameters of the network.

This type of implicit knowledge processing in artificial neural networks is in contrast to the explicit symbolic knowledge processing in traditional artificial

intelligence. Explicit representationalism could account for only a small part of what we call intelligence, but the rest may have nothing to do with systems of symbols [145]. This idea is shared by both the reactive and connectionist approaches. However, guided by Brooks [27, 28], the reactive advocators tend to deny the need for symbolic representations within the machine as far as possible, and use only some form of representation if it is really necessary. This leads to a subtle and difficult choice in design of the primitive reactive modules and the necessary representations and, as a result, they suffer from the designer's influence to some extent [145].

Unlike reactionists, the connectionists try to develop a new learning theory of implicit representation that does not require explicit processing rules [143]. Without explicit symbols, a kind of implicit knowledge representation and processing arise spontaneously and distributively from the artificial neural networks. By adjusting the structure and synaptic weights appropriately, artificial neural networks act as a powerful and general statistical pattern recogniser that can learn any functional mapping [18]. When ANNs are applied to practical problems, some pre-processing of the input data and post-processing of the output are usually required. However, because the learning of artificial neural networks is based on low level primitives of the weights and neurons, it may avoid some undesirable choices made by a human designer [38]. ANNs have been favoured by many researchers as the controllers for producing autonomous and adaptive behaviours for virtual creatures and robots [38, 49, 63, 64, 105, 115, 120, 173, 178].

MENL is appointed as a decision-maker in the Strategy module of the CBG model to choose appropriate motor actions to execute. The ANN in MENL is therefore responsible for learning the action function $\pi(\mathcal{S}, \mathcal{E}, \mathcal{G})$ as introduced in Section 3.4.2. In addition to the environmental information currently perceived by the CBG sensors, the ANN can also utilise the environmental and body state feedback on executed motor actions and the behavioural motivation to guide its selection of motor actions. The learning of the action function π by the ANN is achieved with the assistance of an evolutionary learning algorithm – *Evolution Strategies*.

4.2.1.2 Evolutionary Artificial Neural Networks

There are several ways to improve the structure and parameters of an ANN. A common method is to train ANNs via a large set of sample data. This is called *supervised learning*. However, in the case of natural behaviour learning, it is rather difficult to employ strictly supervised learning algorithms because the correct system output is not always available or computable [108]. In many cases, only some simple signs which indicate the output effect are obtainable from the environment. Therefore, a suitable way to train an ANN for learning natural behaviours may be *self-supervised learning*, that is, to let the ANN learn by itself the correct behaviours through continuous interaction with the environment.

Evolutionary algorithms are good candidates to train an ANN for this type of self-supervised learning because they do not require a direct specification of the desired values for network outputs at any given moment. Instead they use a fitness function to specify a measure of overall performance. Among evolutionary algorithms, Genetic Algorithms (GAs) are widely used to evolve neural control structures [38, 49, 50, 69, 115, 178]. However, recent studies have disclosed that another evolutionary algorithm, *Evolution Strategies (ESs)* have obtained better results than genetic algorithms in many real valued parameter optimisation [10, 130, 136, 137, 141]. Therefore, in the study reported here, we adopt Evolution Strategies to train the artificial neural network of MENL.

Evolution Strategies

Evolution Strategies (ESs) are especially designed for applications that involve real-valued parameters [130, 141]. The solution vector $\vec{x}$ is usually represented as a string of n floating point values, each of which represents one of the objective variables. Together with each value is a control parameter σ that determines the characteristic mutation size for that variable. There may also be an extra control parameter, rotation angle α, which allows mutations to be correlated and the axes of the problem to be rotated arbitrarily. Hence, an individual $\vec{a}$ in a generation of evolution strategies can be represented as $\vec{a} = (\vec{x}, \vec{\sigma}, \vec{\alpha})$.

Evolution strategies use search operators such as mutation and/or recombi-

nation to generate new solutions, and use a selection scheme to test which of the newly generated solutions should survive to the next generation. Historically mutation was the primary move operator in evolution strategies, which is effected by the addition of Gaussian noise (Gaussian mutation) to each objective variable, with a standard deviation controlled by the relevant σ. More recently, a variety of recombination operators have been introduced to either produce one new individual from two random parent individuals or to allow components to be taken for one new individual from potentially all individuals available in the parent population.

Presently, the two most widely used ES algorithms are $(\mu + \lambda) - ES$ and $(\mu, \lambda) - ES$, which are distinguished by their different selection mechanisms. The former selects the best μ individuals from both the μ parents and λ offspring as the parents of the next generation. The latter selects the best μ parents only from the λ offspring. It is believed that the $(\mu, \lambda) - ES$ outperforms the $(\mu + \lambda) - ES$ because $(\mu, \lambda) - ES$ is less likely to land in local optima [10]. The numbers of parents and offspring are recommended to be at a ratio of $\mu/\lambda \approx 1/7$ [10]. (See [10, 11] for further details.)

A key concept of ESs is that both the objective variables $\vec{x}$ and the strategy parameters $(\vec{\sigma}, \vec{\alpha})$ are evolved during the search, exploiting an implicit link between an appropriate internal model and good fitness values. This *self-adaptation* mechanism of strategy parameters allows ESs to self-adapt to different fitness landscapes. In consequence, ESs have no other parameter that has to be tuned by the designer apart from the population size.

In MENL, ESs are used as the main engine to evolve ANN variables (i.e., weights and biases). However, instead of selecting the optimal individual from the last generation and using it as a unique solution, MENL keeps the whole generation of evolution and treats each individual in it as an active agent to make co-decisions.

4.2.1.3 Multiagents

An agent is a system that tries to fulfil a set of goals in a complex dynamic environment [94]. Humans and animals are at the high end of being an agent, with numerous small agents for multiple senses, multiple actions, and complex control structures. Each small agent has also its own specific competence [27, 107] to make contributions to the whole agent system. In a multi-agent system, agents coordinate their knowledge, goals, skills and plans jointly to take actions or solve problems collectively [22]. These agents may work toward separate but related goals, or toward a single global goal. We presume even a simple behaviour may involve a small agent society, in which many small homogeneous agents cooperate and compete with each other to perform the same type of function. When we walk, for example, sometimes we can walk quickly without deep consciousness, but sometimes we may hesitate and clearly have some different ideas conflicting in the brain. If one idea, controlled by an agent, is strong enough to cope with the current situation, the movement quickly resumes. Otherwise, intensive competition or even revision of the agents (e.g., further learning) is evoked.

When the multi-agent system is introduced into the evolution of ANNs, each individual ANN in a generation is an agent. These agents have the same goal but make their own decisions when environmental information is received. While one agent makes only one decision at an instant, multiagents provide more kinds of solutions. Even when the individual which has the highest fitness fails, others with useful ideas can still take over the situation and the behaviour would not be "paralysed". Since multiagents make use of all the population information that is no less than the information held by any single individual, they have enhanced robustness and generalisation ability. In [179], Yao and Liu made use of the population information in evolutionary ANNs to improve generalisation of learned systems. They utilised linear combinations to integrate different individuals in the last generation of evolution to form integrated systems. In contrast to these artificially combined systems, a whole population of agents is maintained in MENL and their relationship is coordinated autonomously and naturally by themselves. (In Section 4.4.2, we will see multiagents compete and complement

with each other to form a coalition for implementing a designated task.) Furthermore, MENL multiagents are not fixed at the last generation of evolution. They are flexible and subject to evolution whenever the situation is out of control. This evolutionary learning is kept through the whole life of an evolutionary ANN.

4.2.1.4 Lifetime Learning

In evolutionary computation, most research work tends to wait for an optimal individual or an optimal generation which satisfies all the pre-designed requirements. Once such a solution appears, it is used as an ideal scheme to deal with all the other problems and won't be improved any more. The ability of the final solution to solve problems then becomes a criterion to consider whether a learning algorithm is successful or not. However, it may be questioned whether an omnipotent solution can always be found, especially in a dynamically changing world. When simulating natural behaviours, it is more important for a solution to be general enough to provide reasonable performance in varied situations, rather than just optimal in some special cases. Even if a solution is optimum on many occasions, there is no guarantee that it will be suitable in other unfamiliar and unusual situations. This solution should therefore be continuously improved so as to suit more and more situations.

Learning is often thought to be equal to optimising an error function or maximising a fitness function. However, we consider learning to be in fact different from optimisation because the learned system should have best generalisation, while the optimised system has best specific competence [115, 179]. A general ability may not be optimal at all, but it should be suitable and adjustable to a wide range of situations. Such an ability is usually accumulated through constant learning in varied situations and this learning will never cease. Those abilities that obtain some functionality but do not update again may face a serious maintenance problem when the situation is not the one they are familiar with.

In the learning algorithm of MENL, a batch of evolutionary agents is maintained. Contrasting to traditional evolutionary computation, these agents are not unchangeable at the last generation of evolution when the evolutionary learning

is finished. Instead, the agents in MENL are flexible and can update themselves through another period of evolutionary learning whenever they encounter problems. This can have a natural bearing on lifetime learning. For example, when the best decision made by the multiagents fails to make a valid action, an evolutionary strategy is then immediately introduced and a new generation of agents is evolved. New decisions are produced and the best one of these is executed again. The new agents will survive the predicament and will be used to control the next situation if a desired result is obtained. Otherwise, further evolution proceeds until new qualified agents are evolved. As this evolutionary procedure iterates during the whole life of MENL, multiagents will have a more enhanced and general ability. This kind of learning is called *lifetime learning* in our study.

Learning continuously in changing environments is also pursued by *lifelong learning* [157] and *anytime Learning* [24, 60, 121]. Lifelong learning studies learning in the context of a long-living robot, which faces collections of learning tasks. An explanation-based neural network learning algorithm (EBNN) is designed to integrates inductive neural network learning and analytical explanation-based learning so the robot can exploit synergy between related learning tasks and transfer knowledge from previously encountered learning tasks to other new learning tasks. This makes the robot particularly applicable to a whole collection of learning tasks over its entire lifetime. Different from lifelong learning which learns tasks one after another, anytime learning learns a particular task and improves its performance continuously. In anytime learning framework, there are usually two systems running in parallel: a *learning system* that learns behaviours in simulated environments, and an *execution system* that uses the best controller produced by the learning system so far to implement behaviours in real environments. At the same time, a *monitor* module in the execution system checks whether the simulated model still matches the real environments, and if not, notifies the learning system to update its simulated model. The lifetime learning of the MENL algorithm is similar to anytime learning in improving the performance of executing a behaviour continuously. However, different from anytime learning, MENL's lifetime learning does not require any simulated models because it learns

behaviours running on actual enviroments. Instead of an engine seperated from the execution system, lifetime learning is indeed part of the execution system and its commands are immediately executed by the execution system. The resulted learning can therefore reflect instant changes in the environment quickly and efficiently.

Due to the utilisation of multiagents and lifetime learning, MENL can attempt to learn a great number of generalisations of behaviours for a wide range of situations. The following exploration experiments in Section 4.4 have shown that evolutionary agents with lifetime learning outperform non-improved multiagents, while multiagents surpass the single agent with the highest fitness.

4.2.1.5 Relaxed and General Fitness Function Design

Usually people think the job of designing a rational fitness function is quite delicate and laborious. In order to channel the evolutionary process in a desired direction, many constraints are craftily imposed on the fitness function. Although these designed fitness functions may obtain some desirable results, they restrict the autonomy of evolution [98] and the generalisation of the solution [50, 179] at the same time. With regard to natural evolution, we find that nature does not enforce any particular criterion on living organisms. It discovers competencies because they can enhance reproductive success. Nevertheless, this happens without specific constraints. As a consequence, it is not surprising that complex and general behaviours are difficult to obtain through artificial evolution confined by many conditions. Recent study on evolutionary learning has confirmed this viewpoint and suggested that behaviours with more complexity could emerge by reducing the constraints imposed on the fitness function and by increasing the affordances of the environment [50].

In order to release the designer from laborious design work, MENL adopts a relaxed fitness function that sketches the effects of decisions made by each agent. This fitness function takes into account the sensory feedback on motor actions and the behavioural objectives. Unlike the previous work of fitness function design that imposes extra constraints to achieve some specific behaviour effects,

such as obstacle avoidance, straight-line walk, etc., the fitness function adopted here employs environment and body state feedback to shape certain aspects of behaviours.

So, the fitness function Φ of an agent $\vec{a}_k$ at time t is:

$$\Phi(\vec{a}_k, t) = \sum_{l=0}^{t} (Response(\Theta(\vec{a}_k, l)) + Aim(\Theta(\vec{a}_k, l))) \tag{4.1}$$

where Θ is the action selection function of agent $\vec{a}_k$.

The first term in the fitness function, *Response*, is the sensory feedback of the motor action $\Theta(\vec{a}_k, l)$ decided by agent $\vec{a}_k$ at time l. The feedback includes corresponding environmental responses and body changes. Instead of being an exact value representing the contribution of an action to a designated behaviour, such as those used in reinforcement learning, the sensory feedback used here is just a simple and general indication of whether an action is good in relation to other actions. For example, when learning space occupying behaviours, the sensory feedback only suggests that successful movement is better than direction change, and direction change is better than no movement which is in turn better than the movement causing collisions. This feedback is a kind of general information common to the behaviours of the same kind (i.e. space occupying movement). It is also easy to observe and obtain.

The second term of the fitness function, *Aim*, indicates the behaviour objective of a particular behaviour. For simplicity, it can be a negative term that indicates how far it is to the behaviour objective at the present time. This term is mainly used to encourage an agent to act for a designated behaviour, and not to be lost in some other meaningless actions. As a result, when assisted by the general sensory feedback to behaviours of the same kind, MENL can learn to perform particular behaviours with the guidance of specific behaviour objectives.

The fitness function Φ is a sum of all the effects of actions $\Theta(\vec{a}_k, l)$ chosen from the initial time $l = 0$ to the present moment t. The fitness function hence not only reflects the effect of the current decision made by an agent, but also takes into account the decision-making history of the agent.

When selecting a decision of actions to execute, those agents that may produce the best effect in the current situation are considered first. Past experience is

the second factor to affect selection. So, if more than one agent selects the best action(s) at an instance, the one that has the highest fitness value (i.e., the best experience) will take priority to control the current situation. This design derives from an attempt to balance the relationship between generalisation and optimisation. By only thinking of an immediate action and not making a comprehensive consideration of past experience, it is possible to produce a local optimum result suitable for limited occasions. On the other hand, considering past experience too much and not focusing on the current situation may cause a mediocre solution that implements a task at a low level of competence. Here, we try to balance these two factors by providing a general but also capable framework of action selection. Further study on this topic is our future work.

4.2.2 Work Engine of MENL

Through the above introduction of the essential strategies used in MENL, it should be clear how MENL works in general. Table 4.1 gives an algorithmic description of MENL. The concrete working procedure of MENL for achieving a behavioural motivation can be described as follows.

When MENL starts, a population of evolutionary ANN individuals $P(s) = \{\vec{a}_1(s), ..., \vec{a}_\mu(s)\}$ at generation $s = 0$ and time $t = 0$, is created. The population consists of μ individuals, $\vec{a}_k \in I = R^n \times R^n_+, \forall k \in \{1, ..., \mu\}$. Every individual $\vec{a}_k$ is composed of an n-dimensional vector $\vec{x}_k$ that is the object variable component to be learned, and an n-dimensional vector $\vec{\sigma}_k$ which is the mutation size component. For simplicity, MENL does not adopt the rotation angle α. Each individual in a population is actually an artificial neural network, which is also a MENL agent. The object variables in $\vec{x}$ of each individual are therefore the weights and biases of the neural network this individual represents.

After the population creation, MENL checks if the goal $\mathcal{G}$ of the behaviour to be achieved is reached. The goal can be that the virtual creature has arrived at a designated location in the goal reaching behaviour, or that a maximum time is reached in the exploration behaviour. If the goal of the behaviour is reached, the work of the MENL algorithm then stops. Otherwise, MENL should

1. t=0; s=0;
2. initialise $P(s) = \{\vec{a}_1(s), ..., \vec{a}_\mu(s)\} \in I^\mu$
   where $I = R^n \times R^n_+$,
   and $\vec{a}_k = (\vec{x}_k, \vec{\sigma}_k), \quad \forall k \in \{1, ..., \mu\}$;
3. if (goal $\mathcal{G}$ is achieved) goto step 15;
   else goto step 4;
4. make decisions on $P(s) : \{\Theta(\vec{a}_1(s), t), ..., \Theta(\vec{a}_\mu(s), t)\}$
   where $\Theta(\vec{a}_k, t) = ANN(\vec{a}_k, \mathcal{V}, t)$, that is, the output of an ANN at time t, whose parameters come from $\vec{a}_k$ and input is $\mathcal{V}$ (e.g., the currently perceived environmental information);
5. evaluate $P(s) : \{\Phi(\vec{a}_1(s), t), ..., \Phi(\vec{a}_\mu(s), t)\}$
   where $\Phi(\vec{a}_k, t) = \sum_{l=0}^{t} Effect(\Theta(\vec{a}_k, l))$
   and $Effect(\Theta) = Response(\Theta) + Aim(\Theta)$;
6. select the best decision $\Theta_{best} = \Theta(\vec{a}_j, t)$ which has $max(\Phi(\vec{a}_j, t))$,
   $j \in \{k | max(Effect(\Theta(\vec{a}_k, t)))\}$, $k \in \{1, ..., \mu\}$;
7. execute Θ_{best}
   t=t+1;
8. if $(Response(\Theta_{best}) == penalty)$ goto step 9;
   else goto step 3;
9. recombine: $\vec{a}'_k(s) = r'(P(s)) \quad \forall k \in \{1, ..., \lambda\}$;
10. mutate: $\vec{a}''_k(s) = m'_{\{\tau,\tau',\beta\}}(\vec{a}'_k(s)) \quad \forall k \in \{1, ..., \lambda\}$;
11. make decisions on $P''(s) : \{\Theta(\vec{a}''_1(s), t), ..., \Theta(\vec{a}''_\lambda(s), t)\}$;
12. evaluate $P''(s) : \{\Phi(\vec{a}''_1(s), t), ..., \Phi(\vec{a}''_\lambda(s), t)\}$;
13. select: $P(s+1) = s_{(\mu,\lambda)}(P''(s))$;
    s=s+1;
14. goto step 3;
15. stop.

Table 4.1: The algorithmic description of MENL

make decisions about what action to take next. Based on the input information $\mathcal{V}$ at time t (e.g., the perceived environmental information about the present situation), all of the current agents existing in MENL simultaneously make their own decisions of action selection for achieving the behaviour. Each agent $\vec{a}_k$, which is also a neural network individual, uses its network parameters to calculate a decision output, $\Theta(\vec{a}_k, t) = ANN(\vec{a}_k, \mathcal{V}, t)$, which is a mapping from the input information at time t to a decision on actions. All of the output solutions are evaluated by the fitness function Φ. Among those output solutions suggested by the multiagents, the one finally chosen Θ_{best} to instruct the behaviour should have the best effect $max(Effect(\Theta(\vec{a}_k, t)))$ in the current situation t, and the best decision-making history (the highest fitness Φ) if there are more than one solution producing the best effect. All of the multiagents are kept to control the next situation at time $t+1$ if a positive response is obtained after the execution of the selected solution. However, if a penalty is received, it means that there is no competent agent present at the moment to instruct the behaviour properly. In this situation, evolutionary learning is immediately triggered to search for new agents. The population at generation s is first recombined by the recombination operator r', and produces λ offspring $\vec{a}'_k$, $\forall k \in \{1, ..., \lambda\}$. Then, the offspring are mutated by the mutate operator $m'_{\{\tau,\tau',\beta\}}$, where τ, τ', and β are parameters for generating the mutation size $\vec{\sigma}$. The best μ evaluated individuals in the offspring is chosen as new μ parents (multiagents) at generation $s + 1$. New evolved agents make their decisions for the current situation and the best decision is executed. These new agents are maintained if they are able to resolve the situation. Otherwise, even further evolutionary learning is involved until a new generation of qualified agents is obtained. The above procedure iterates until the goal of the designated behaviour is achieved. The last multiagents surviving the behaviour achievement should possess the basic skills for that behaviour. Nevertheless, those agents are still subject to update whenever they cannot work in new situations and/or for achieving new behaviours.

When MENL makes decisions on action selection for a particular behaviour, the perceived information, such as that of the environment, is an important in-

put to MENL. In addition, whenever a decision made by MENL is executed, some responses from the outside (e.g., environment) are obtained, which indicate how good the decision is. These responses are useful for determining whether the current multiagents are competent in the current situation. The responses are also essential to the evolution of agents when these agents are incompetent. Because responses are recorded by term *Response* in the fitness function and accumulated from the beginning to the current time, they actually compose part of the decision-making history of agents. This history provides a useful guidance to guide the evolutionary learning to move towards the successful achievement of the behaviour, together with the behavioural objective also encoded in the fitness function. In this sense, MENL is much more than a reactive controller that purely reacts to the environment. Indeed, MENL can be goal-oriented because it can select actions according to behavioural objectives, and this selection can be continuously improved based on the sensory feedback and past experience. As the perceived environmental information constitutes the direct input to MENL, the information about the sensory feedback and past experience is the implicit knowledge stored in the artificial neural network and evolutionary learning of MENL. Based on the perceived and collected information, MENL can learn to make correct decisions of action selection for achieving single and multiple behaviours.

4.2.3 Adaptive Learning of MENL in the CBG Model

When the MENL learning algorithm is introduced into the CBG model, we mainly use it in the Strategy module of the CBG model to learn the correct motor action selection policies. Therefore, MENL is responsible for the implementation of the module "Make decisions" in the adaptive strategy shown in Figure 3.6. The CBG module provides MENL with the necessary information (e.g., perceived environmental information, body states of the virtual creature, and collected sensory feedback), and executes the motor actions that MENL selects. In turn, MENL should learn the correct action function $\pi(\mathcal{S}, \mathcal{E}, \mathcal{G})$ mapping each input triplet $(\mathcal{S}, \mathcal{E}, \mathcal{G})$ into a selection of candidate actions in $\mathcal{A}$ (see Section 3.4.2).

The learning of the action function π is achieved by the multiagents in MENL. From the above description of the MENL work engine, we see that every agent in MENL has an output function Θ that translates its input information into a motor action selection output. In terms of both the explicit and implicit input information involved in the translation, this output function is actually the action function π_k of a MENL agent, as the goal of the behaviour to be achieved is $g \in \mathcal{G}$, and the body states $s_t \in \mathcal{S}$ and the environmental information $e_t \in \mathcal{E}$ at time t provide the explicit information about the environment and the body states and the implicit information about sensory feedback on executed motor actions. The decision of action selection made by agent $\vec{a}_k$ at time t is therefore:

$$\Theta(\vec{a}_k, t) = \pi_k(s_t, e_t, g) \tag{4.2}$$

Among the decisions made by multiagents, the actual action chosen by MENL and then executed by the CBG model is the result produced after two extra selection procedures. The first selection procedure f_1 chooses the decisions that have the best effects in the current situation for achieving the behaviour g. The second selection procedure f_2 then selects the one that has the highest fitness from those decisions preferred by the first selection. So, the actual action decided by MENL at time t is:

$$\pi(s_t, e_t, g) = f_2(f_1(\pi_k(s_t, e_t, g))) \qquad \forall k \in \{1, ..., \mu\} \tag{4.3}$$

where π is the action function of MENL and π_k is the action function of agent $\vec{a}_k$ in MENL.

As multiagents make their own decisions on action selection, the real action determined is a sensible selection on these decisions. Therefore, the action function of the MENL learning algorithm is a result selected from the multiple choices provided by the multiagents in MENL. Because the individual action function of every agent is continuously evolved through evolutionary learning, the resulting action function of MENL is constantly improved as well.

In Chapter 3, we introduced the notion that the CBG model provides the Strategy module with the basis for adaptive learning, allowing the Strategy module to learn from past useful or harmful experience. How then, does the MENL

learning algorithm utilise this mechanism to adjust its action selection policies when it is embedded in the Strategy module of the CBG model?

The adaptive learning of MENL in the CBG model is achieved by the self-adjustment of multiagents (ANN individuals) via evolutionary learning. The exploitation of lessons from unsuccessful experience is obvious: when the CBG model detects from its sensory feedback that an unexpected result is generated, MENL will be informed of such information immediately. Evolutionary learning is then triggered in MENL to correct the mistake by evolving new qualified multiagents. The memorisation and utilisation of successful experience is automatically achieved in MENL. Once a correct decision of action selection has been made and successfully executed, the current multiagents are kept to deal with subsequent situations. The useful decision-making skills held by these multiagents are therefore maintained and can be reused in the future. Even if the agents are evolved when they fail to generate correct decisions in some other occasions, the valuable experience obtained by these agents can still be retained for the following generations through evolution. The new generated multiagents therefore have not only new learned skills from their own practice, but also useful knowledge inherited from their parents. Due to the continuous self-adjustment, the CBG combined with MENL can have more and more enhanced action selection ability and can cope with more and more situations during its life. The following experiments on exploration learning of the CBG combined with MENL have supported this claim.

4.3 Implementation of MENL for Space Occupying Behaviours

In Section 3.7, we introduced how to set up the Computational Behaviour Generation model for achieving space occupying behaviours. When the MENL learning algorithm is applied in the Strategy module of the CBG model to learn the suitable action selection policies for space occupying behaviours, the implementation details of MENL are as follows.

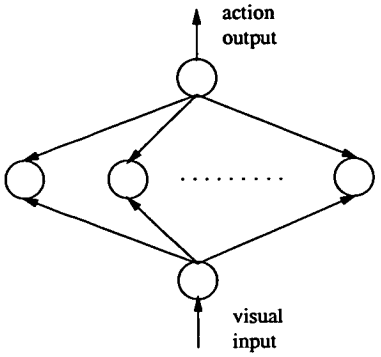


Figure 4.1: The feedforward artificial neural network in MENL

4.3.1 ANN Structure

The artificial neural network adopted in MENL is a simple feedforward neural network with one hidden layer (Figure 4.1). Ten hidden neurons are used to connect one visual input and one action output.

The input to MENL is the visual information of the current situation which is collected by the synthetic visual sensor of the CBG model as described in Section 3.7.1. The output of the network consists of a single component that selects one of the eleven motor actions of the virtual creature (see Section 3.7.5.1 for the explanation of motor actions).

In addition to the visual information input, the information such as the behavioural objective and sensory feedback on executed motor actions is also used to guide the decisions of motor action selection of the ANN. This information, as explained above, is introduced into the fitness function of the evolutionary artificial neural network to evolve the action selection policies.

4.3.2 Evolution Strategies

Evolution strategies, as introduced in Section 4.2.1.2, are the main engine to evolve the ANN in MENL. As recommended in [141], (15,100)-ESs are adopted here. That is, at each step of evolution, 15 parents survive from 100 offspring and act as the current multiagents. For simplicity, only standard deviation σ is used

and no rotation angle α is presented. The initial strategy parameter (standard deviation) is set to 0.5. The initial objective parameters (weights and biases of the ANN) are small random numbers in the interval [-1,1]. In evolution, strategy parameters are operated on by discrete recombination and objective parameters by panmictic intermediate recombination.

The fitness function of an evolutionary individual, also a MENL agent $\vec{a}_k$, consists of the sensory feedback *Response* and the behavioural objective *Aim* (see Section 4.2.1.5). Because the eleven motor actions (moving forward in three different directions, turning to the other seven directions other than the current one, and doing nothing) produce four types of results in an environment: successful walk, direction change, no movement and collision, four kinds of environmental responses to motor actions are accordingly designed. These environmental responses are reward to successful movement (reward_move), reward to direction change only (reward_dir), reward to no action (reward_fix), and penalty to movement causing collisions (penalty_collision). In addition, when the body states indicate that the creature is detained at a place for a long time by several direction changes or no movement, the State module of the CBG model will report another two penalty responses: penalty to unuseful direction changes (penalty_dir) and penalty to no actions (penalty_fix). Therefore, the sensory response function $Response(\Theta(\vec{a}_k, t))$ to an action Θ decided by agent $\vec{a}_k$ at time t has six possible values: reward_move, reward_dir, reward_fix, penalty_dir, penalty_fix, and penalty_collision. As the reward responses generate an appraisal on the decision of actions, the penalties indicate a harmful decision that should be corrected via evolutionary learning. All the responses may have arbitrary values [1], but satisfy the following condition:

$$reward\_move > reward\_dir > reward\_fix > penalty\_dir > penalty\_fix > penalty\_collision \quad (4.4)$$

The behavioural objective *Aim* in the fitness function indicates the behavioural motivation of the virtual creature, which the CBG model should achieve. When

[1]The response values can be positive or negative, as long as they satisfy the condition 4.4.

simulating space occupying behaviours, the behavioural objective is always represented by a negative item. In exploration, it is the accessing variable *Access* of an environmental place to be accessed by agent $\vec{a}_k$, which implies how many times the creature has been in this place before [2]. The accessing variable for a location is incremented by one each time it is visited. So, combining the sensory responses *Response* and the behavioural objective *Access* together, the fitness function Φ for exploration is expressed as:

$$\Phi(\vec{a}_k, t) = \sum_{l=0}^{t}(Response(\Theta(\vec{a}_k, l)) - Access(\Theta(\vec{a}_k, l))) \tag{4.5}$$

In the behaviour of goal reaching, the sensory feedback to motor action decisions is the same as that in exploration. However, the behavioural objective of goal reaching is different. This objective is represented by a negative item, $Dis(\Theta, Goal)$ in the fitness function, which is the distance of the place about to be accessed by an agent from the goal destination. Guided by this item, an agent can move towards a place nearer to the goal destination. The fitness function for the goal reaching behaviour is therefore:

$$\Phi(\vec{a}_k, t) = \sum_{l=0}^{t}(Response(\Theta(\vec{a}_k, l)) - Dis(\Theta(\vec{a}_k, l), Goal)) \tag{4.6}$$

In the behaviour of wandering, a clear motivation is absent. The potential purpose is only to move around in an environment without causing collisions. This is mainly controlled by the sensory feedback. So there is only sensory feedback present in the fitness function for wandering, shown by the following equation:

$$\Phi(\vec{a}_k, t) = \sum_{l=0}^{t}(Response(\Theta(\vec{a}_k, l))) \tag{4.7}$$

[2]Recent study on magnetic sense has suggested that some animals may use the magnetic field of the earth as not only a possible cue for move orientation and/or distance, but also a potential source of world-wide positional information [89, 90, 165]. Here, we assume the virtual creature has a similar simulated magnetic sense to determine its position and the distance to a goal. Because the State module records the position information when executing a space occupying behaviour, the virtual creature can easily use the State module to find out the access times of a place. The information (place position and distance between two places) can actually be obtained from the graphical system of the virtual environment.

In Section 3.7.5.1, we mentioned that both common and different factors are involved in the action selection for generating various space occupying behaviours. These factors are considered in the fitness functions of multiagents. As the term *Response* is the common feedback obtained from the environment and body states after action execution, the term *Aim* takes different factors into account to reflect different behavioural motivations. Guided by the common and different factors, multiagents in MENL are evolved to learn the correct action selection policies for implementing different but functionally related behaviours.

4.4 Experiments

We have introduced the basic ideas of the CBG model and the MENL learning algorithm, and the implementation details of the CBG combined with MENL for achieving space occupying behaviours. In this section, we report some experiments to test the adaptive learning ability of the CBG combined with MENL. Specifically, a virtual creature is equipped with the CBG with MENL to learn the exploration behaviour. The creature should access places in an unknown environment as much as possible within a limited period. In addition, the creature should carry out obstacle avoidance during its exploration. The main purpose of the experiments is to examine whether the creature equipped with the CBG with MENL can learn exploration in various unknown environments and how adaptive its learning ability is. Some hand-crafted exploration experiments are also designed separately for an evaluation of the learning performance of the CBG with MENL.

4.4.1 Experimental Setup

In the exploration learning experiments, the virtual creature is put into various unknown environments to learn the exploration behaviour. Every exploration experiment is repeated for fifty runs. In each run, the virtual creature needs to explore a test environment many times so as to learn the proper exploration in this environment. Every exploration results in an exploration trajectory in the

environment. Each trajectory, starting from an arbitrary position, only lasts for a certain period, which is a count of the number of actions executed by the creature. An exploration trajectory is said to be "successful" if there is no collision or evolutionary learning involved. Otherwise, a trajectory is said to be "unsuccessful" or "failed". One run of learning of exploration within an environment is finished only when the virtual creature conducts one hundred successful trajectories successively. In this case, we think the creature has grasped the basic exploration skills in this environment. The learning performance averaged over fifty runs of each experiment is shown in the following figures.

Three kinds of performance indices are recorded in every exploration trajectory for a test of the learning performance. The indices adopted here are an external performance metric, instead of an internal performance metric such as the fitness function. Internal measures of performance do not always coincide with external measures of performance: sometimes the internal metric indicates that a controller's performance is improving, while the external metric denotes it is decreasing [176]. In order to evaluate the CBG and MENL's actual performance, we utilise the external measure in this thesis, which comes from an external observer's estimate of performance.

For the three performance indices, the first one records the number of places accessed by the virtual creature in an exploration trajectory as a percentage of the number of free places in the environment. This index is called *exploration efficiency*. The second index, *collision times*, is the number of collisions the creature makes in a trajectory. The third index records *learning times*, that is, how many times the multiagents have been evolved in a trajectory. The evolutionary learning is due to the wrong actions selected and executed which cause unexpected penalties. As the virtual creature explores an environment more and more times, we expect that the creature gradually learns how to select correct motor actions to execute. In consequence both the learning times and collision times spent in each exploration trajectory should decrease and the exploration efficiency of each trajectory should increase.

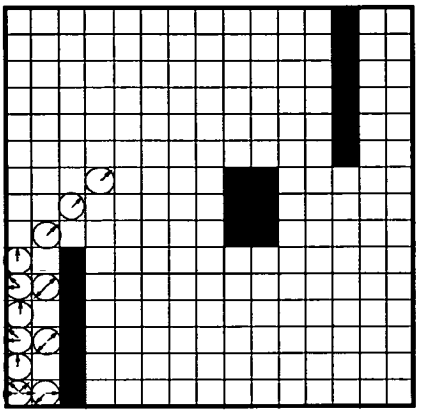


Figure 4.2: Environment E1 and an exploration starting from the bottom left corner

4.4.2 Exploration Learning in Various Unknown Environments

In the experiments, the virtual creature equipped with a randomly initialised CBG and MENL is first put into a simple environment to learn basic exploration skills. It is then introduced into other more complicated environments to test its adaptation ability further. During the experiments, the creature is also put back into simpler environments after learning in more complicated environments to investigate whether the creature still remembers how to explore in an earlier environment and whether learning in more complex environments implies knowledge of simpler environments. All the environments used in the test are completely unknown to the creature.

4.4.2.1 Fresh Learning in a Simple Environment E1

A simple environment E1 (Figure 4.2) is chosen as the first testing ground for exploration. There are three big obstacles present in this environment, represented as black boxes in Figure 4.2. A *fresh* virtual creature that is furnished with an initialised CBG and MENL is used to conduct *fresh learning* of exploration in E1. In each run of fresh learning, the creature is trained in the environment E1 via trajectories starting from various initial positions that are all randomly selected. In order to give the virtual creature plenty chances to explore the envi-

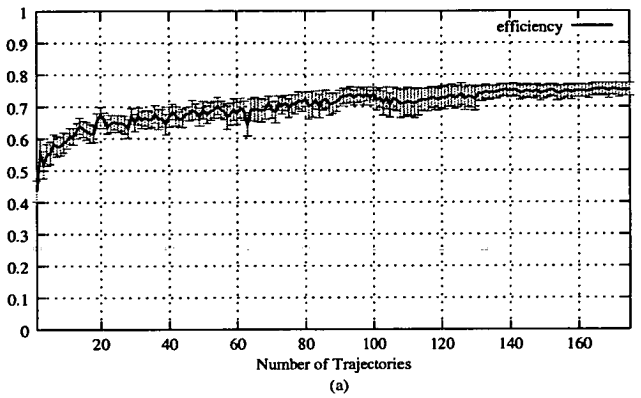
ronment, every trajectory of exploration consists of two hundred and forty steps, i.e., two hundred and forty actions executed by the virtual creature. This number is about twenty percent more than the number of spare places in the environment E1, which is one hundred and ninety five. The creature executes exploration trajectories one after another until one hundred successive trajectories have been executed successfully. The recorded learning results are shown in Figure 4.3 [3], which are averaged over fifty runs of this experiment.

In Figure 4.3, Figure 4.3 (a) shows the exploration efficiency of each exploration trajectory, which indicates how many places are visited in this trajectory. Figure 4.3 (b) shows the number of evolutionary learning procedures and collisions made by the creature in each trajectory. Figure 4.3 (c) lists the total learning times and collision times of all trajectories, which are required in the fresh exploration learning, and the exploration efficiency the creature achieves at the end of the learning.

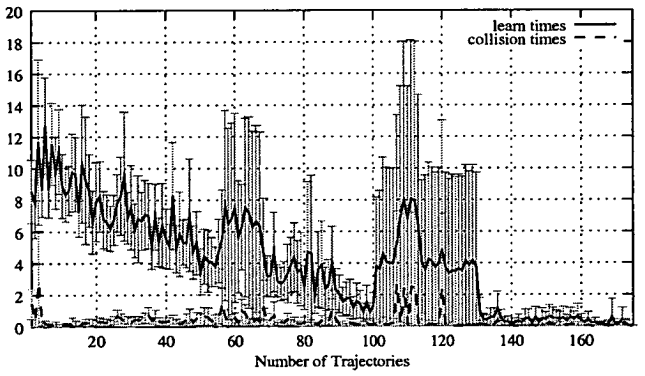
As expected, the virtual creature has gradually grasped the exploration skills in E1 through adaptive learning. The learning has taken a maximum of 173 trajectories to reach success in fifty runs. The exploration efficiency obtained in each trajectory obviously increases as the creature explores the environment E1 (see Figure 4.3 (a)). When learning starts, the fresh creature cannot explore the environment very well: it accesses 85 squares on average in the first trajectory, which are only 43.6 percent of all the free squares in E1. However, the virtual creature visits more and more squares in each trajectory thereafter, by using new learned policies of motor action selection. As a result, the exploration efficiency of each trajectory gradually improves. When learning finishes, the virtual creature can produce exploration so smoothly that there is no longer any collision or evolutionary learning generated. The exploration efficiency finally achieved by the creature is 75.7%, about 32.1 percentage points higher than the efficiency obtained at the beginning of the learning.

Along with the increasing exploration efficiency, the learning procedures and

[3]In this thesis, the experimental data shown in the result graphs are the means of the fifty runs. The error bars show the 95% confidence intervals. See Appendix A for detailed calculations.



(a)

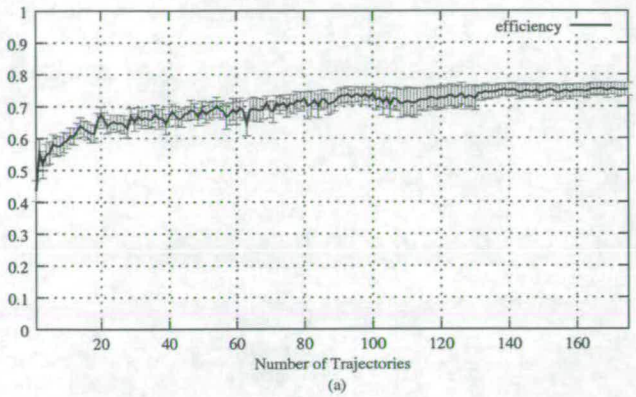


(b)

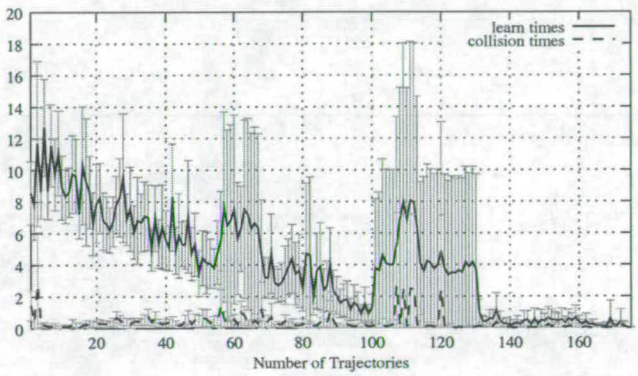
Total learning times	Total collision times	Final efficiency
726.8 (523.1-930.5)	56.7 (34.5-79.0)	75.7% (75.5%-75.9%)

(c)

Figure 4.3: Fresh exploration learning in environment E1 (averaged over fifty runs). In each run, a fresh virtual creature furnished with a randomly initialised CBG and MENL conducts exploration trajectories in the environment continuously, until one hundred successive and successful exploration trajectories have been executed. Every exploration trajectory starts at different positions and lasts two hundred and forty steps. The exploration efficiency reached by each trajectory is shown in Figure (a). The learning times and collision times made in each trajectory are shown in Figure (b). Both figures show results averaged over fifty runs and do not include the last 100 successive and successful trajectories. Figure (c) is a summary of the total learning times and collision times spent in fresh learning and the exploration efficiency finally achieved by the fresh learning, averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.



(a)



(b)

Total learning times	Total collision times	Final efficiency
726.8 (523.1-930.5)	56.7 (34.5-79.0)	75.7% (75.5%-75.9%)

(c)

Figure 4.3: Fresh exploration learning in environment E1 (averaged over fifty runs). In each run, a fresh virtual creature furnished with a randomly initialised CBG and MENL conducts exploration trajectories in the environment continuously, until one hundred successive and successful exploration trajectories have been executed. Every exploration trajectory starts at different positions and lasts two hundred and forty steps. The exploration efficiency reached by each trajectory is shown in Figure (a). The learning times and collision times made in each trajectory are shown in Figure (b). Both figures show results averaged over fifty runs and do not include the last 100 successive and successful trajectories. Figure (c) is a summary of the total learning times and collision times spent in fresh learning and the exploration efficiency finally achieved by the fresh learning, averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

No. of steps	Creature positions	Sensor input	Actions chosen	Update agents
1	(0,14,2)	0.331	10	no
2	(1,14,2)	0.221	10	yes
3	(1,14,2)	0.221	6	no
4	(1,14,7)	0.11	11	no
5	(0,14,6)	0	8	no
6	(0,14,5)	0.214	7	no
7	(0,14,3)	0.655	3	yes
8	(0,14,5)	0.214	11	no
9	(0,13,4)	0.469	11	no
10	(1,12,3)	0.503	5	no
11	(1,12,7)	0.441	11	no
12	(0,12,6)	0	11	yes
13	(0,12,5)	0.214	11	no
14	(0,11,4)	0.552	11	no
15	(1,10,3)	0.669	5	no
16	(1,10,7)	0.524	11	no
17	(0,10,6)	0	11	yes
18	(0,10,5)	0.214	11	no
19	(0,9,4)	0.607	11	no
20	(1,8,3)	1	10	no
21	(2,7,3)	0.986	10	no
22	(3,6,3)	0.979	10	no

Figure 4.4: An example of the CBG combined with MENL at work. The virtual creature starts to explore E1 from the bottom left corner and faces to the east, as shown in Figure 4.2. Creature position (x, y, d) indicates the current square location the creature occupies along the horizontal (from left to right) and vertical (from top to bottom) axis, and the current view direction of the creature (see Figure 3.9 for the meaning of direction labelling). Actions $1 \sim 8$ correspond to no movement and direction changes to the other 7 directions. Actions 9, 10 and 11 refer to movements to the left forward, straight forward, and right forward.

collisions required in each trajectory are decreasing (see Figure 4.3 (b)). When the virtual creature starts to learn exploration, there are a few collisions and many learning procedures involved in each trajectory. Nevertheless, the average number of learning procedures rises and falls and is kept below one after 131 trajectories. At the same time, the average number of collisions is approaching null. When the experiment finishes, there are no evolutionary learning or collisions present at all. The total numbers of learning times and collision times spent in fresh exploration learning are 726.8 and 56.7 respectively, averaged over fifty runs.

Figure 4.2 shows a small part of the exploration trajectory where the virtual

creature starts from the bottom left corner. Figure 4.4 illustrates how the CBG combined with MENL works in this trajectory. This figure shows the sensory information and the motor actions chosen at each time step. It also indicates whether the multiagents should be evolved after the execution of the motor actions.

At the first step, the virtual creature is placed at the bottom left corner (labelled as (0,14) with the original set at the top left corner), and faces to the east (i.e., direction 2 according to the direction labelling shown in Figure 3.9). Accordingly the sensors of the creature feed an input value 0.331 to the CBG model, and the multiagents of MENL make their own decisions of what action to execute based on this input. Of 15 decisions, 11 of them is to change to a different direction. Three other decisions are to move to the right forward, which cause the creature to crash into the wall. There is only one agent that chooses motor action 10, which is to move straight forward. Since this action will safely move the creature to a new spare square in the environment, it becomes the winner of the agent decisions. This action is then executed by the CBG model and the creature is now moved into a new place (1,14).

In the new situation, the creature directly faces a big obstacle. The sensory input value is therefore changed to 0.221. Based on the new input, the multiagents make decisions to move either straight forward, left forward or right forward for the next movement. Because the last agent which correctly moved the creature into the new place obtains a high reward in the fitness, its current decision is chosen to execute again, which is also to move straight forward. However, while the creature tries to move one step forward, it actually bumps against the big obstacle. This collision reflects a penalty from the environment, indicating the failure of the multiagent decisions. In consequence evolution strategies are used to evolve the multiagents in order to seek new competent agents to solve the current problem. After evolutionary learning, another generation of the multiagents is generated and these new agents make new deicisions on motor action selection based on the input value 0.221. This time, a direction change motor action is selected which changes the view direction of the creature to the southwest, i.e.,

direction 7.

After the direction change motor action is executed successfully, the creature receives a new input value of 0.11. Based on the new input, most agents still choose motor actions of direction change, except one agent chooses to move straight forward and another one decides to move to the right forward. Because movement to the straight forward will result in a collision, movement to the right forward which does not collide takes control of the creature's movement. Therefore the creature moves back to the bottom corner (0,14) but faces to new direction 6.

Now the visual field of the creature is fully blocked by the wall, so all of the multiagents decide to change the view direction. The agent which made the correct decision last time wins the competition due to its higher fitness. The creature is then turned to direction 5. Next the creature is turned to direction 3 and direction 5 based on the motor actions chosen by the same agent. Even though there is no collision made during the movement, the sensors of the creature detects that the creature stays in the same place (0,14) for too many steps, and the multiagents should be updated so that the creature can access more other free places in the environment. In consequence evolutionary learning is employed again to evolve a new generation of the multiagents. After evolution, new generated multiagents choose to move right forward from the bottom left corner, and accordingly the creature steps into a new place (0,13) with direction 4. Following this movement, the multiagents continue their successful decision of the right forward movement. As a result the creature successfully accesses another place of environment E1, which is place (1,12).

In the new place, the creature meets the big obstacle again and the sensory input becomes 0.503. While all of the movements to the left forward, straight forward, and right forward would cause a collision, the multiagents choose a motor action of direction change. The view direction of the creature is therefore changed to the southwest. The new direction brings new free places in the visual field, so a movment to the right forward is decided by the multiagents and executed by the motors. The creature now moves into the new place (0,12) with direction 6.

In the new location the multiagents insist on a move to the right forward, and as a result, the creature changes to direction 5 but hits the wall. The hit results in a penalty from the environment and accordingly a new evolution is made of the multiagents. Located at place (0,12) and facing northwest, new agents choose to move to the only free square (0,11) in the current visual field. The right forward movement is carried out again so the creature moves to the square (1,10) with view direction 3. Faced with the big obstacle, motor action 5 is again chosen by the multiagents so as to change the view direction of the creature. After the direction change, the multiagents still decide to move right forward, and therefore, the creature moves to a new location (0,10) with direction 6.

In the new location, the multiagents choose the movement to the right forward again. Therefore the creature turns direction to its right but cannot move forward anymore. Since the attempt to move forward results in a collision with the wall, the multiagents are updated by evolutionary learning for a correct decision in the new situation. After evolution, the multiagents decide to move to the free places (0,9) first and (1,8) next. Now with an open visual field full of free squares, multiagents make several easy decisions of moving straight forward. These decisions finally help the creature move away from the big obstacle.

From Figure 4.2 and Figure 4.4, we can see that, even though the creature is initially "puzzled" by the walls and obstacles and is almost lost in the bottom left corner, it at last steps out of the impasse after many sessions of evolutionary learning. The creature then moves smoothly thereafter.

Multiagents vs. the Best Agent

As we introduced in Section 4.2.1, multiagents in MENL co-decide decisions on motor action selection for the creature to carry out. After the creature has learned how to explore the unknown environment E1 in fresh exploration learning, we are eager to know what contributions the multiagents make to exploration, and whether the best agent alone is competent in successful exploration. Bearing these questions in mind, we command the virtual creature with the learned multiagents and a creature with the best of the multiagents, i.e., that which has the highest

	Efficiency achieved	Collision times	Learning times
Multiagents	74.3% ~ 77.8%	0	0
Best agent	0.51 ~ 35.6%	0 ~ 240	0

Table 4.2: Comparison of multiagents and the best agent in their decision-making ability of suitable motor action selection (averaged over fifty runs). Both the group of learned multiagents after fresh exploration learning in E1 and the best agent of the group are used to explore E1 for an extra ten times but with no more learning. This table lists the exploration efficiencies and the numbers of collisions of those ten trajectories, averaged over fifty runs.

fitness, to execute an extra ten exploration trajectories starting from random squares. Neither the multiagents nor the best agent are learning via evolutionary strategies any more. This experiment is repeated 50 times, and the average results are shown in Table 4.2.

In ten exploration trajectories, the multiagents perform quite well compared with the best agent. The multiagents do not hit any obstacles, and access 145 to 152 squares, which are 74.3% to 77.8% of the free squares in E1. It appears that the multiagents have acquired a good exploration ability. Contrasting with the good performance of the multiagents, the best agent alone is not able to explore the environment E1 competently. The best agent can only traverse 69 squares at most in one trajectory, resulting in a low exploration efficiency of 35.6%. In the worst case, the best agent cannot move around since it starts the exploration by insisting on moving forward when facing a big obstacle. Therefore, the original square where it is put is the only place that it visits, and the number of collisions can be up to 240. The best agent is often unable to free itself when encountering a difficulty.

In multi-agent exploration, there is usually more than one agent contributing to the action selection, rather than only the best one. Figure 4.5 shows an example of the contribution times of each agent in one experimental run, averaged over

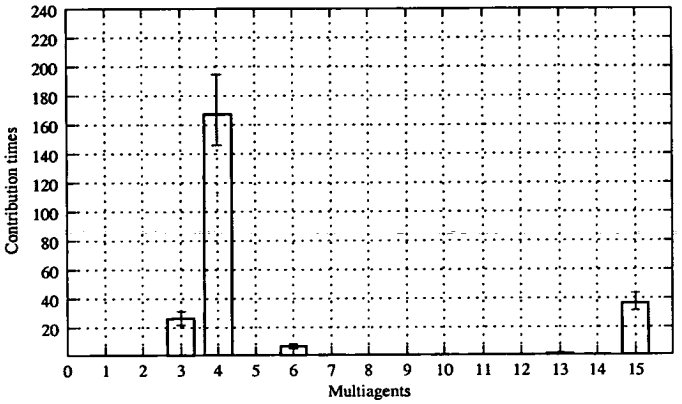


Figure 4.5: Contribution times of multiagents in one experimental run (averaged over ten trajectories). Multiagents compete and cooperate with each other to conduct ten random exploration trajectories successfully. The contribution times indicate how many times the decisions of action selection made by one agent are finally chosen to be executed by the virtual creature.

Action No. / Agent No.	2	3	5	9	10	11
3			3.3	23.1		
4	9.4	0.2		58	99.4	
6					6.7	
7					0.1	
13					2.3	
15						37.5

Table 4.3: Selection times of each motor action by contributing agents in one experimental run (averaged over ten trajectories). Blank cells in this table indicate zero selections have been made. Actions 1 $\sim$ 8 correspond to no movement and direction changes to the other 7 directions. Actions 9, 10 and 11 refer to movements to the left forward, straight forward, and right forward. Actions and agents not listed have no contribution in this run.

ten trajectories. The contribution times indicate how many times the decisions of action selection made by one agent are finally chosen to be executed by the virtual creature. Table 4.3 shows the corresponding motor actions chosen by those contributing agents to explore the environment. Of the fifteen agents, there are mainly six agents contributing to exploration. The fourth agent is the best agent and controls the most situations in one trajectory (in average 167 situations in 240 steps). However, there are still some situations that the best agent cannot deal with. Once the best agent fails, it cannot escape from its difficulties by itself. Fortunately, some other agents can still have different but useful decisions to take charge of these difficult situations. These agents include the 3rd, 6th, 7th, 13th and 15th agents. Although these agents cannot generally supersede the best one, they may save the virtual creature on some occasions. Because different agents with different structures are good in varied situations, actions co-decided by multiple agents are more robust than those decided by only the best one. A successful exploration is indeed accomplished by the cooperation of a whole generation of agents.

It is interesting to see that, of the choice of eleven possible motor actions, several different actions are favoured by different agents (see Table 4.3). Agent No.4, for instance, prefers to move straight forward and left forward in most cases, or to turn to two other different directions occasionally. The third agent, nevertheless, can also decide to move left forward and change to a third direction when agent No.4 fails to make a correct action selection in some situations. Agents 6, 7 and 13, share the straight forward movement with the fourth agent in minor situations. Agent 15 then undertakes the entire responsibility of the right forward movement. As a result, the whole exploration is implemented by not only the competition but also the cooperation in the multiagents. This cooperation naturally arises from the autonomous, adaptive learning of multiagents, instead of being designed by a human designer beforehand. Although some agents (those not listed in Table 4.3) have no contribution in this experimental run, they are maintained with contributing agents. The population information held in all of the fifteen multiagents can therefore be used or transferred to their offspring as

a whole for implementing future tasks.

Figure 4.6 and Table 4.4 present another experimental run in which multiagents exhibit varied contributions to exploration. This time, seven agents contribute to the finally selected motor actions. Similarly to the first example, the movement of going forward is the most favoured by several agents, whilst the left forward movement, right forward movement, and direction changes are used to assist the exploration. Due to the presence of multiple agents, the exploration is executed smoothly and successfully.

4.4.2.2 Successive Learning in a More Complicated Environment E2

The creature that learned successfully in E1 is put into another unknown environment E2 (Figure 4.7) to test if it can explore in a more complicated environment. Environment E2 is a larger environment than E1 and possesses several obstacles which have different sizes and shapes. The exploration is therefore more difficult than that in E1 because the situations the virtual creature would meet are more varied and the creature has to execute correction actions in response to various visual input. In E2, a trajectory of exploration starts from an arbitrary position and consists of 650 steps which is about 20% more than the number (552) of the spare squares. The creature should conduct exploration trajectories continuously until one hundred consecutive and successful trajectories are executed. Every creature of the previous fifty runs of fresh exploration in E1 is used to continue the exploration in E2. The average results over the fifty runs in E2 are shown in Figure 4.8. Similar to the previous experiment, these results record the average exploration efficiency, collision times and learning times of every trajectory.

Because the environment E2 is a new environment different from the previously traversed environment E1, the creature encounters many new difficulties that it has not met before. The creature inevitably makes collisions when it uses the old exploration skills to deal with new situations. However, due to the lifetime learning maintained in multiagents, the creature learns to overcome these difficulties through continuous evolutionary learning. As Figure 4.8 (b) shows, there are some collisions in early trajectories of the successive learning, but the

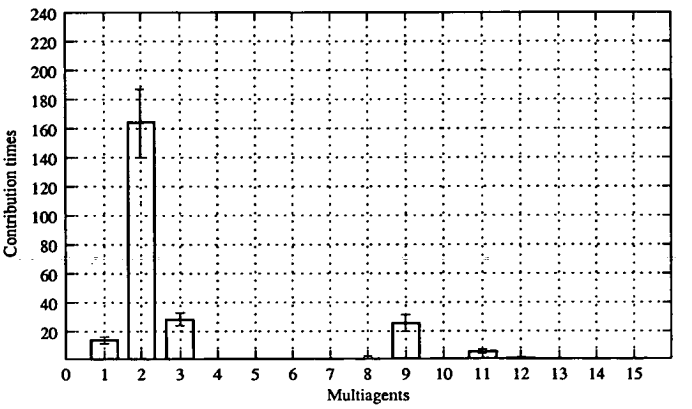


Figure 4.6: Contribution times of multiagents in another experimental run (averaged over ten trajectories). Multiagents compete and cooperate with each other to conduct ten random exploration trajectories. The contribution times indicate how many times the decisions of action selection made by one agent are finally chosen to be executed by the virtual creature.

Action No. / Agent No.	2	5	9	10	11
1			13		
2		4.8	54.9	104.7	
3	4.9			5.9	17.3
8				1.4	
9				21.1	4.1
11					5.9
12					2

Table 4.4: Selection times of each motor action by contributing agents in another experimental run (averaged over ten trajectories). Blank cells in this table indicate zero selections have been made. Actions $1 \sim 8$ correspond to no movement and direction changes to the other 7 directions. Actions 9, 10 and 11 refer to movements to the left forward, straight forward, and right forward. Actions and agents not listed have no contribution in this run.

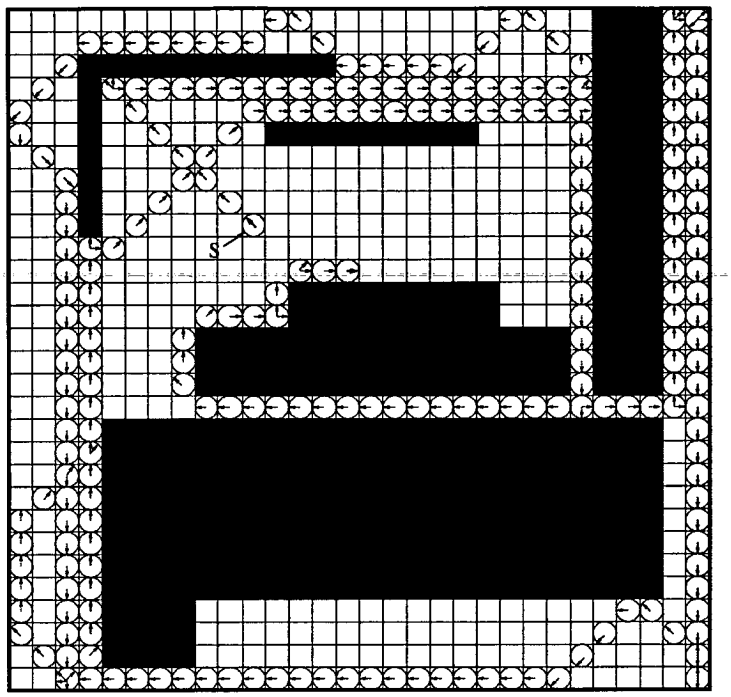
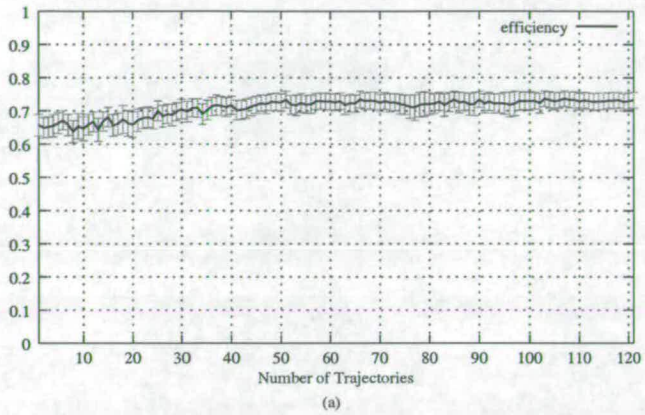
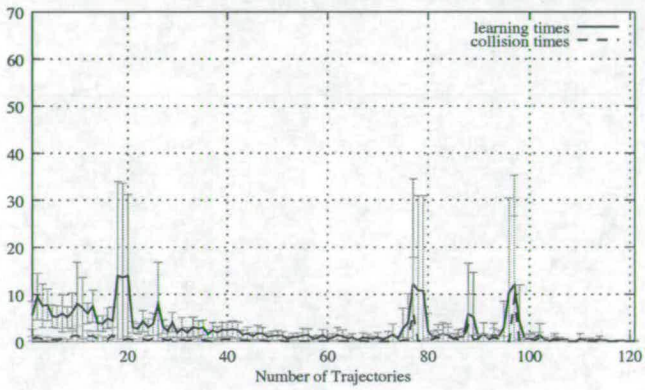


Figure 4.7: Environment E2 and an exploration starting from a random place S

number of collisions is eventually reduced to zero after 103 trajectories. The total number of collisions the creature makes for learning successful exploration in E2 is about 56.7, averaged over fifty runs. In order to overcome collisions and learn to explore E2, there are also some evolutionary learning procedures involved in the experiment. However, all the creatures in fifty runs have learned exploration successfully in a maximum of 121 trajectories. They execute smooth exploration without collision or learning after successful learning. In total, 348.7 sessions of evolutionary learning are required for learning exploration in E2, averaged over fifty runs. As the creature learns exploration further, it accesses more and more places in each trajectory. The exploration efficiency is therefore continuously improved. The efficiency is about 65.7% at the beginning of the experiment, but increases to 73.3% when the learning finishes. The virtual creature has adapted to the new environment E2 through its continuous interaction with the environment.



(a)



(b)

Total learning times	Total collision times	Final efficiency
348.7 (128.5-569.0)	56.7 (0-116.9)	73.2% (73.0%-73.4%)

(c)

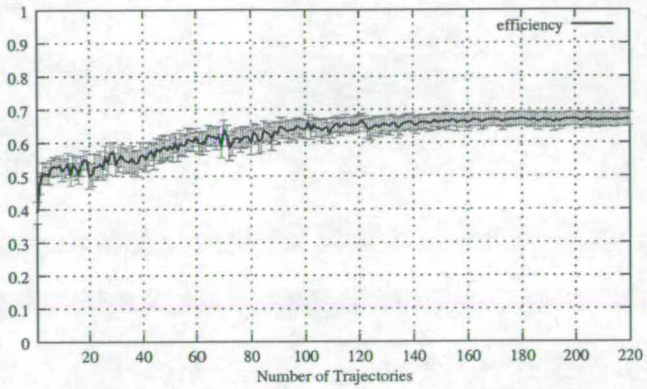
Figure 4.8: Successive exploration learning in environment E2 (averaged over fifty runs). Every creature which has learned from the previous fifty runs of fresh exploration in E1 is asked to explore a new unknown environment E2. The creature should conduct continuous exploration trajectories in E2 until one hundred successive and successful trajectories are executed. An exploration trajectory in E2 starts from an arbitrary position and lasts 650 steps. Figure (a) records the exploration efficiency achieved by each trajectory in the successive learning in E2, and Figure (b) shows the learning times and collision times made in each trajectory. Both figures show results averaged over fifty runs and do not include the last 100 successive and successful trajectories. Figure (c) lists the total learning times and collision times, and the exploration efficiency finally achieved in the successive learning, averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

Accumulated Exploration Skills

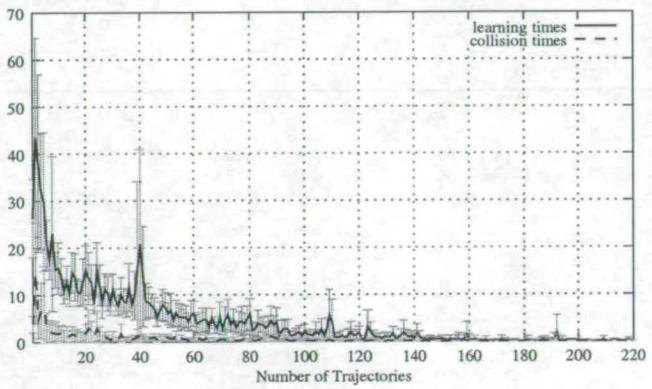
By comparing the successive learning in E2 with fresh learning in E1, we have found that learning in E2 is even faster and easier than that in E1: not only are fewer learning procedures required in E2 than in E1, but the exploration efficiency also starts at a higher rate in E2. This fact is somewhat surprising because environment E2 is actually more complicated and bigger than environment E1. In order to test if the previous learned exploration ability in E1 has any effect on the successive exploration in E2, a fresh virtual creature equipped with a randomly initialised CBG with MENL is asked to conduct exploration learning in E2 separately. This new creature is commanded to execute arbitrary exploration trajectories in E2 until it conducts successful trajectories for 100 successive times. Fresh learning in E2 is executed for fifty runs and the average results are shown in Figure 4.9.

Through the experimental results of fresh learning in E2, we see that the fresh virtual creature has also learned the exploration ability in E2 successfully. The creature requires a maximum of 220 trajectories to learn the exploration in fifty runs. For fresh learning in E2, the creature has tried to access more and more places whilst avoiding obstacles. Both learning and collision times gradually decrease via evolutionary learning. Meanwhile, the exploration efficiency increases from 39% to 67.1%.

However, the fresh creature's overall performance in E2 is worse than that of the creature with successive learning. Compared with successive exploration, fresh learning in E2 involves many more collisions and learning procedures, which are on average 131.6 and 964.3 respectively. In addition, while the successive exploration learning starts from a high exploration efficiency (65.7%), fresh exploration learning in E2 only starts from 39%. Even when fresh learning finishes, the traversed places are still about 6 percentage points less than those traversed in the successive learning. Contrasted with learning from scratch, successive learning in E2 is much enhanced and accelerated through the accumulated experience obtained from early exploration learning in E1. The exploration skills transferred from E1 to E2 have helped the learning in E2 to be much easier and more efficient.



(a)



(b)

Total learning times	Total collision times	Final efficiency
964.3 (818.2-1110.3)	131.6 (86.2-177.1)	67.1% (66.9%-67.3%)

(c)

Figure 4.9: Fresh exploration learning in environment E2 (averaged over fifty runs). In each run, a fresh creature equipped with an initialised CBG and MENL is used to explore environment E2. Every exploration trajectory in E2 starts at an arbitrary position and lasts 650 steps. The exploration efficiency reached by each trajectory is shown in Figure (a). The learning times and collision times made in each trajectory are shown in Figure (b). Both figures show results averaged over fifty runs and do not include the last 100 successive and successful trajectories. Figure (c) lists the total learning times, the total collision times, and the exploration efficiency finally achieved in fresh learning, averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

Evolutionary Multiagents vs. Non-improving Multiagents

In Section 4.4.2.1, multiagents have demonstrated correct action selection ability and successful exploration in E1. However, these multiagents have had to learn further so as to adapt to new situations in E2. What then, would the exploration be if the multiagents explore the environment E2 without learning? Are they still able to cope with the new environment through their coordination?

In order to test the exploration ability of non-improving multiagents, a virtual creature with the learned multiagents from environment E1 is put into E2 to conduct ten random exploration trajectories. This time, there is no lifetime learning kept in those multiagents so the multiagents cannot be improved by learning even when they encounter difficulties. The experimental results averaged over fifty runs are shown in Table 4.5.

In ten exploration trajectories, the places explored by non-improving multiagents range from 0.6 to 37.7 percent of the free places in environment E2. Those agents cannot always avoid collisions properly and the collision times could be up to 504 in a 650-step trajectory. Indeed, non-improving multiagents are not able to execute smooth exploration in E2. A common case in the experiment is that non-improving multiagents usually can move around for some steps based on their co-operation, but they will find themselves stuck in a predicament once they fail to make a suitable movement to avoid a big obstacle or move out from a corner. Without lifetime learning, non-improving multiagents cannot improve themselves to cope with new difficulties.

In contrast to non-improving multiagents, evolutionary multiagents have shown much more flexibility and adaptation in their exploration in E2, as shown in the results above (Figure 4.8). Although multiagents with lifetime learning also meet difficulties in their exploration, they are able to learn to generate new solutions to overcome these difficulties.

Figure 4.7 shows a typical exploration trajectory in which the creature is initially placed at a random place S. This trajectory is a good example to illustrate how evolutionary multiagents use their lifetime learning to handle difficult situations. In this trajectory, the virtual creature starts its exploration from a lo-

	Efficiency achieved	Collision times	Learning times
Non-improving Multiagents	0.6 ~ 37.7%	0 ~ 504	0

Table 4.5: Exploration of non-improving multiagents in E2 (averaged over fifty runs). The multiagents learned from environment E1 are used to explore the new environment E2 with no more learning. These agents conduct ten random trajectories in E2 and this experiment repeats for fifty runs. This figure lists exploration efficiencies and collision times made by non-improving multiagents in ten trajectories, averaged over fifty runs.

cation S. It walks forward and soon gets into the corner of the Γ-shaped obstacle. The multiagents with limited exploration ability acquired from fresh learning in E1 always attempt to move straight forward but all crash into the obstacle. If the multiagents are not improved, the exploration can only end with 7 squares accessed in environment E2.

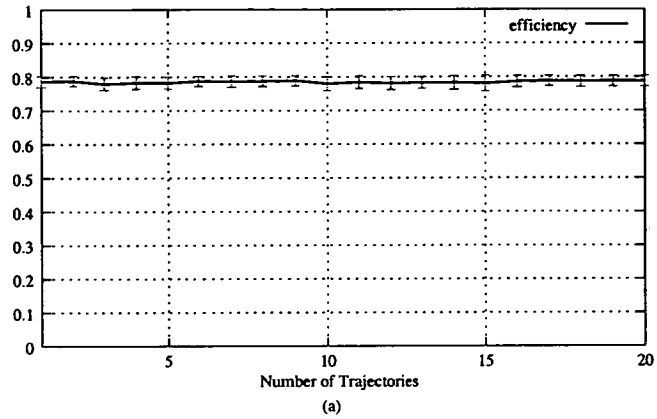
When multiagents learn through evolutionary learning during their lifetime, they continuously improve their abilities through trial and error. After 7 movements, the virtual creature walks into the Γ-shaped corner and hits the obstacle for the first time. In this situation, an evolutionary strategy is immediately triggered to evolve the multiagents. After a short period of self-adjustment, new agents decide to turn to the right and hence move out from the corner successfully. When the virtual creature with new agents executes the traverse from the original place S again, it stops at the Γ-shaped corner. Instead of moving forward, the creature turns to the right first and then moves forward. This time, there is no collision made at all. Evolved multiagents have not only learned how to deal with difficult situations, but also remember the learned skills so as to process these difficulties properly in the future. Due to their continuous updating ability, evolutionary agents with lifetime learning have presented much better exploration performance than non-improving agents.

Re-exploration in Environment E1

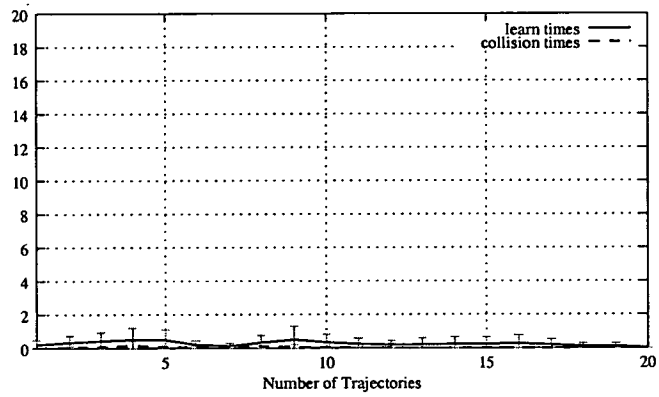
The successive training in E2 after fresh exploration in E1 has shown that the virtual creature can be trained to explore both environments E1 and E2 competently. However, it is not clear whether the creature can still perform well in the earlier environment E1 after the sequential training in E2. For the purpose of test, we put the creature into environment E1 again to investigate whether it still remembers the earlier learned exploration skills in E1. Therefore every creature of the previous fifty runs of successive exploration in E2 is used to explore environment E1 for a second time. Similar to the fresh learning in E1, each trajectory in re-exploration includes 240 steps and starts from an arbitrary place. This exploration continues until one hundred consecutive and successful trajectories are executed. The average results over fifty runs in the second exploration of E1 are shown in Figure 4.10.

The experimental results show that the virtual creature after sequential learning can still explore environment E1 quite well. There are only very occasional evolutionary learning and collisions made in some exploration trajectories. In fifty runs, thirty six experimental runs have no learning or collisions present at all. That is, the virtual creatures in these runs move in environment E1 freely with no problem. All of the creatures in the remaining fourteen runs have collected their previously learned exploration skills in E1 in a maximum of 20 trajectories. The total numbers of evolutionary learning and collisions involved in the re-exploration are in average about 5.1 and 0.7 respectively.

When the creature explores environment E1 for a second time, it is worth noting that the exploration efficiency is clearly improved. During the re-exploration of E1, the exploration efficiency is kept around 78.7%, which is 3 percentage points more than that achieved in the fresh exploration. After learning a new environment E2, the creature can still explore the previously learned environment E1 competently.



(a)



(b)

Total learning times	Total collision times	Final efficiency
5.1 (1.1-9.1)	0.7 (0-2.1)	78.7% (77.3%-80.1%)

(c)

Figure 4.10: Re-exploration in E1 (averaged over fifty runs). Every creature which has been trained in E1 and E2 sequentially in the previous fifty runs is asked to explore the environment E1 again. The creature conducts continuous exploration trajectories in E1 until one hundred successive and successful trajectories are executed. An exploration trajectory in E1 starts from an arbitrary position and lasts 240 steps. Figure (a) records the exploration efficiency achieved by each trajectory in the second exploration in E1, and Figure (b) shows the learning times and collision times made in each trajectory. Both figures show results averaged over fifty runs and do not include the last 100 successive and successful trajectories. Figure (c) lists the total learning times and collision times, and the exploration efficiency finally achieved in the successive learning, averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

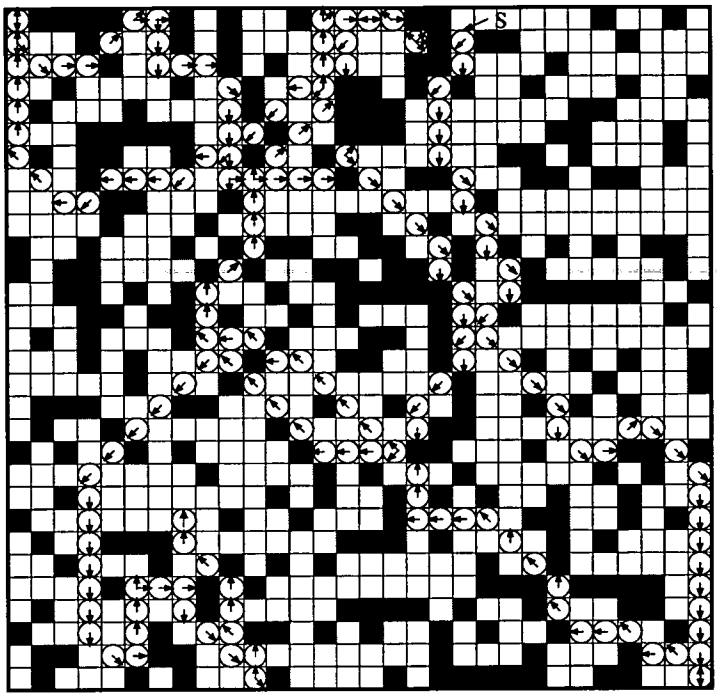
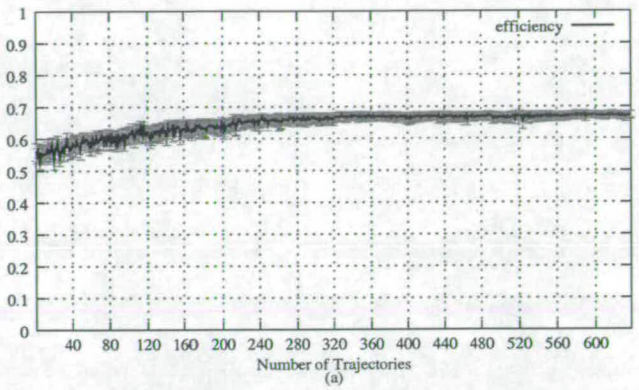


Figure 4.11: Environment E3 and an exploration trajectory starting from a random place S

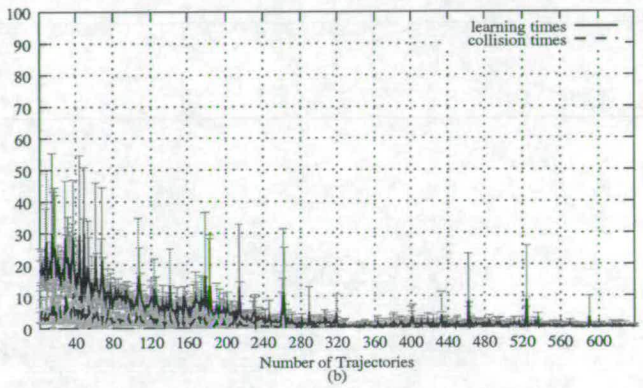
4.4.2.3 Successive Learning in a Maze E3

The third testing ground for the exploration test is a maze E3, shown in Figure 4.11. All the obstacles in the maze are randomly generated. This new environment E3 is much more complicated than the previous two environments E1 and E2. Even worse, the obstacles and their arrangement in E3 are not regular and have no clear rules to follow. The virtual creature faces a great challenge of adapting to its new environment, after exploring in E1 and E2.

When exploring E3, the virtual creature starts from arbitrary places and orientations and explores E3 for 780 steps in each trajectory. This number is again about 20% more than the number (630) of free places in the environment. The experiment is finished when the creature has successfully explored the maze for one hundred consecutive trajectories. The creatures resulting from previous fifty runs of successive exploration in E2 (also from the fresh exploration in E1) are ap-



(a)



(b)

Total learning times	Total collision times	Final efficiency
2877.9 (2150.4-3605.2)	758.2 (667.9-948.5)	67.9% (67.8%-68.0%)

(c)

Figure 4.12: Successive exploration learning in maze E3 (averaged over fifty runs). The creature having learned from E1 and E2 is asked to learn exploration in E3. Each creature resulting from the previous fifty runs of successive exploration in E2 (also from fresh exploration in E1) continuously conducts exploration trajectories starting from random positions in E3 until it explores the environment successfully for one hundred successive times. Each exploration trajectory in E3 consists of 780 steps. Figure (a) shows the exploration efficiency achieved in each trajectory, and Figure (b) shows the learning times and collision times made in each trajectory, all averaged over fifty runs. The last 100 successive and successful trajectories are not shown in Figure (a) and (b). Figure (c) lists the total learning times, the total collision times, and the exploration efficiency finally achieved in successive learning in E3, averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

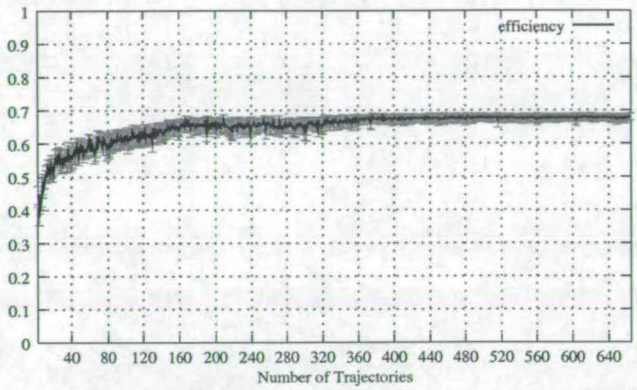
plied to conduct another fifty runs of successive learning in E3. The experimental results averaged over those fifty runs are shown in Figure 4.12.

As expected, the virtual creature suffers from great difficulties during its exploration in E3. It has made many collisions and much evolutionary learning accordingly. The average number of collisions is around 6 in each trajectory at the beginning of the experiment, and the average number of learning procedures is around 18. However, both learning and collision times decrease when learning progresses. At the same time, the exploration efficiency gradually increases. When the creature explores the environment E3 for a maximum of 638 trajectories in fifty runs, it has learned how to explore E3 successfully. The average total number of evolutionary learning procedures used in the experiment is 2877.9, and the average total number of collisions made is 758.2. When successive learning in E3 finishes, the creature can accesses 67.9% percent of the free squares in E3.

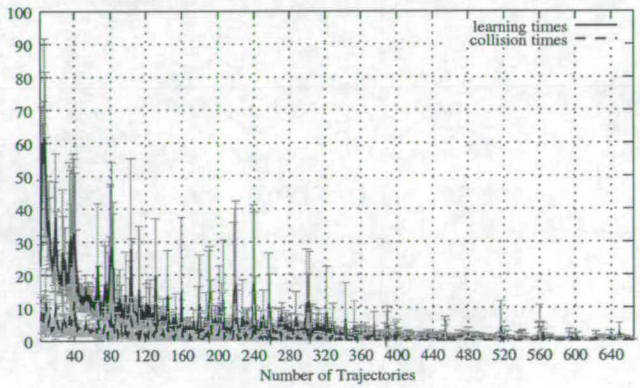
Accumulated Exploration Skills

A fresh creature with an initialised CBG and MENL is asked to conduct exploration learning in E3. Figure 4.13 shows the experimental results over fifty runs. Similar to the successive learning in E3, the fresh creature experiences an intensive learning procedure and a gradually increased exploration ability. However, compared with the creature which has accumulated exploration experience from the previous exploration in E1 and E2, the fresh creature exhibits a poorer learning ability. At the early stages of the experiment, the fresh creature requires almost twice the learning procedures to complete an exploration trajectory of 780 steps. The exploration efficiency is only about 44.1% when the experiment starts, which is ten percentage points lower than that of successive learning. The total number of collisions made by the fresh creature is as high as 883.9 on average over fifty runs and the total number of learning procedures on average reaches 3546.4 when fresh learning finishes. After conducting 651 trajectories, the fresh creature "grasps" the regulation of exploration in E3. The final exploration efficiency comes up to 67.5% at last, which is similar to that of successive learning.

Due to the inherent complexity of the maze E3, both creatures with accumu-



(a)



(b)

Total learning times	Total collision times	Final efficiency
3546.4 (2696.1-4396.8)	883.9 (637.5-1130.4)	67.5% (67.4%-67.6%)

(c)

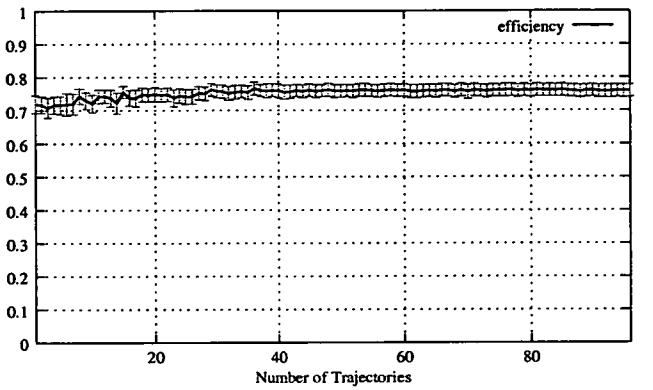
Figure 4.13: Fresh exploration learning in maze E3 (averaged over fifty runs). In each run, a fresh creature with initialised CBG and MENL is used to conduct exploration learning in E3 independently. Each exploration trajectory in E3 starts at an arbitrary position and lasts 780 steps. The exploration efficiency reached by each trajectory is shown in Figure (a). The learning times and collision times made in each trajectory is shown in Figure (b). Both figures show results averaged over fifty runs and do not include the last 100 successive and successful trajectories. Figure (c) lists the total learning times, the total collision times, and the learned exploration efficiency of fresh learning in E3. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

lated exploration experience and with no experience at all encounter great difficulties in their exploration. However, in both cases, the creatures have adapted to the new complicated environment through adaptive learning. Compared with fresh learning, the successive learning in E3 has once again demonstrated that the exploration experience accumulated in simple environments can help the learning of exploration in a complicated environment to be carried out more smoothly and competently, even though the final performance (final efficiency) of the two creatures might be similar.

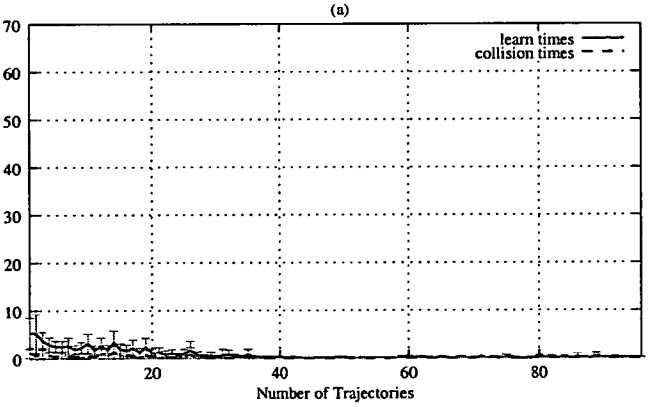
4.4.2.4 Exploration Learning from Complex to Simple Environments

In the above sections, we have tested the exploration learning ability of a virtual creature equipped with CBG and MENL from simple to complex unknown environments. The experiments have shown that the creature can learn to explore these environments successfully, and the learning in simpler environments can help the following exploration in more complex environments. In this section, we conduct another experiment to investigate the learning performance of CBG and MENL from complex to simple environments. In particular, we use what the virtual creature learned from fresh exploration in the most complex environment E3 to explore the less complex environment E2. Therefore, every virtual creature in the fifty runs of fresh exploration in E3 should explore environment E2 continuously until one hundred successive trajectories have been successfully made. Each trajectory in E2 lasts 650 steps and starts from an arbitrary square. The average results over fifty runs are illustrated in Figure 4.14.

From the experimental results, we can see that the creature after fresh exploration learning in E3 can learn to explore the simpler environment E2 quite well. Due to the different environment layout, the virtual creature has spent some evolutionary learning to learn to explore environment E2. However, the learning performance is much better than any other kind of learning in E2 executed before, including fresh learning and successive learning in E2 after the fresh learning in E1. Having obtained exploration skills in the complex environment E3, the learning in a less complicated environment is clearly easier. When the



(a)



(b)

Total learning times	Total collision times	Final efficiency
69.24 (36.5-101.9)	18.1 (4.9-31.3)	76.1% (75.9%-76.3%)

(c)

Figure 4.14: Successive exploration learning in E2 after fresh learning in E3 (averaged over fifty runs). The creature learned from complex environment E3 is asked to learn a simpler environment E2. Each creature resulting from the previous fifty runs of fresh exploration in E3 continuously explores E2 from random positions until it explores the environment successfully for one hundred successive times. Each exploration trajectory in E2 consists of 650 steps. Figure (a) shows the exploration efficiency achieved in each trajectory, and Figure (b) shows the learning times and collision times made in each trajectory, all averaged over fifty runs. The last 100 successive and successful trajectories are not shown in Figure (a) and (b). Figure (c) lists the total learning times, the total collision times, and the exploration efficiency finally achieved in this learning, averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

creature starts to explore the environment E2, it immediately accesses 72% of the free squares in the environment. The exploration efficiency is improved to 76.1% when the learning finishes, which is 9 percentage points higher than the efficiency achieved in fresh learning in E2 and 2.9 percentage points higher than in the sequential learning of E1 and E2. At the same time, the creature has spent much fewer evolutionary learning times and collisions during its successive exploration in E2 after fresh exploration in E3. Over fifty runs, the average number of evolutionary learning procedures spent is only 69.2 and the average number of collisions is only 18.1. This is in constrast with the 964.3 evolutionary learning procedures and 131.6 collisions spent in fresh exploration in E2, and the 348.7 evolutionary learning procedures and 56.7 collisions in the successive learning of E2 after fresh learning in E1. If we take cumulative cost into account, learning E3 then E2 has in total 3615.6 evolutionary procedures and 902 collisions. The number of evolutionary procedures is much lower than that of learning E2 then E3, which is 3925.3. The number of collisions made in learning E3 then E2 is similar to that in learning E2 then E3, which is 897.3. (Due to limited space, we don't introduce the sequential learning of E3 after fresh exploration in E2 here. Interested readers can refer to Appendix B for the relevant experimental results.)

When the virtual creature uses CBG combined with MENL to learn a behaviour continuously, the learned skills of executing this behaviour can be accumulated from environment to environment, both from simple to complex, or from complex to simple environments.

4.4.3 Hand-crafted Exploration in Unknown Environments

For a better understanding of the performance level of the adaptive learning of the CBG with MENL, a controller is hand-crafted to take full responsibility for the decision-making of motor action selection for exploration. Because each time a virtual creature moves it can only move one step, this controller takes advantage of the places in front of the creature in its visual field, by carefully selecting a suitable place to move to. In this thesis, the creature can only detect three places next to it, that is, places 0, 1, and 7 as illustrated in Figure 3.9. So the

availability of these three places is considered when the controller decides which place to move to. The place chosen should be free of obstacle and should have been so far accessed least among the three places in front of the virtual creature. If all the places in front of the creature are full of obstacles, the controller then decides to turn around so as to find an available place to explore. The hand-crafted controller can help us to determine how good an exploration ability a reasonably successful exploration might achieve, and how the adaptively learned exploration ability of the CBG with MENL compares relative to the hard-wired controller.

A new virtual creature starting off with the hand-made controller has executed exploration experiments in environments E1, E2 and E3 respectively. In each environment, the creature conducts one hundred trajectories continuously, and each trajectory starts at a randomly selected position. The trajectories in each environment consist of the same numbers of steps as those in adaptive learning. That is, an exploration trajectory is 240 steps in environment E1, 650 steps in E2 and 780 steps in E3. The experiment in every environment is repeated for fifty runs and the average exploration efficiencies obtained are shown in Figure 4.15.

With the hand designed exploration policy, the virtual creature always selects the best place in front of it to move to so as to avoid obstacles and explore the environments successfully. In consequence the creature accesses on average about 74.9% of the free squares in E1 through 240 steps. In environment E2, the average exploration efficiency reaches about 75.6% in a trajectory. In maze E3, it is 70.5%.

Figure 4.16 is a list of performances achieved by hand-crafted experiments, fresh learning experiments, and successive learning experiments in the three testing grounds E1, E2 and E3. The performances include the final exploration efficiencies achieved in all experiments, and the learning times and collision times required in adaptive learning experiments. From this figure, we can see that, the adaptive learning of the CBG model combined with MENL, especially successive learning, has reached similar exploration efficiencies to those of the hand-crafted controller that is based on the information about the places in front of the virtual

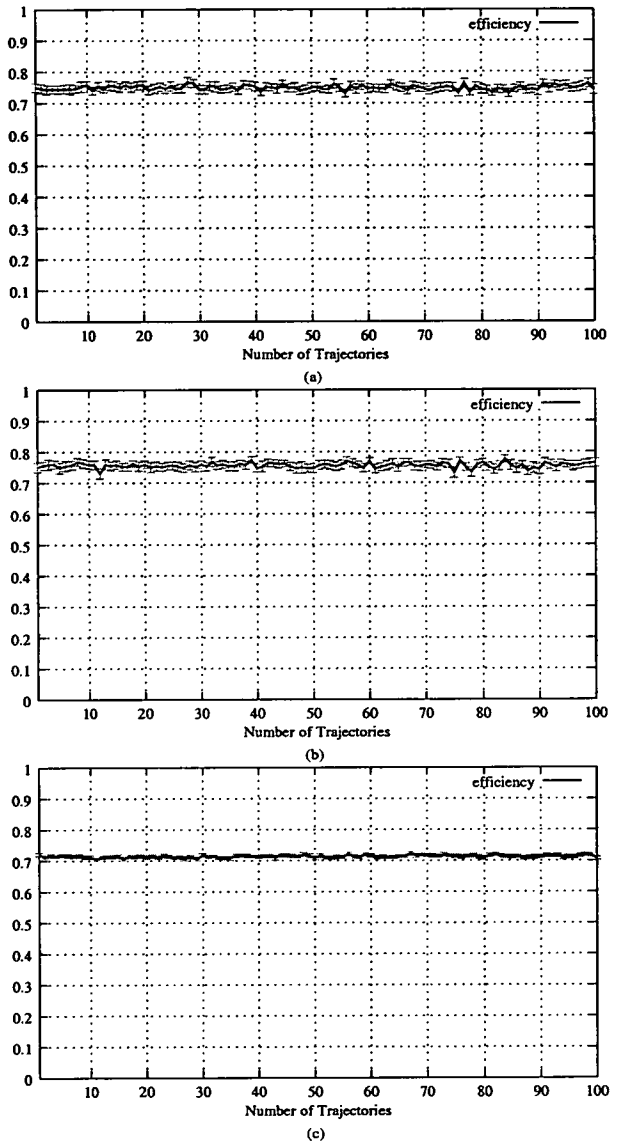


Figure 4.15: Hand-crafted exploration in unknown environments E1, E2 and E3 (averaged over fifty runs). A hand-crafted controller is designed to choose the best motor actions according to the availability information of the places in front of the virtual creature in the visual field. The creature uses such a controller to explore unknown environments E1, E2 and E3 respectively, each for one hundred trajectories. Every exploration trajectory starts at an arbitrary position and consists of the same number of steps as that in adaptive learning, that is, 240 steps in E1, 650 steps in E2, and 780 steps in E3. Figure (a) shows the exploration efficiency of each trajectory in E1, and Figure (b) in E2 and Figure (c) in E3. The results shown in the figures are averaged over fifty runs.

		E1	E2	E3
Final efficiency	Hand crafted	74.9% (74.8%-75.0%)	75.6% (75.5%-75.7%)	70.5% (70.4%-70.6%)
	Fresh learning	75.7% (75.5%-75.9%)	67.1% (66.9%-67.3%)	67.5% (67.4%-67.6%)
	Successive learning	X	73.2% (73.0%-73.4%) (E1 -> E2)	67.9% (67.8%-68.0%) (E1 -> E2 -> E3)
		78.7% (77.3%-80.1%) (E1 -> E2 -> E1)	76.1% (75.9%-76.3%) (E3 -> E2)	X
Learning times	Fresh learning	726.8 (523.1-930.5)	964.3 (818.2-1110.3)	3546.4 (2696.1-4396.8)
	Successive learning	X	348.7 (128.5-569.0) (E1 -> E2)	2877.9 (2150.4-3605.2) (E1 -> E2 -> E3)
		5.12 (1.1-9.1) (E1 -> E2 -> E1)	69.2 (36.5-101.9) (E3 -> E2)	X
Collision times	Fresh learning	56.7 (34.5-79.0)	131.6 (86.2-177.1)	883.9 (637.5-1130.4)
	Successive learning	X	56.7 (0-116.9) (E1 -> E2)	758.2 (667.9-948.5) (E1 -> E2 -> E3)
		0.7 (0-2.1) (E1 -> E2 -> E1)	18.1 (4.9-31.3) (E3 -> E2)	X

Figure 4.16: List of performances achieved by hand-crafted experiments, fresh learning experiments, and successive learning experiments in three environments E1, E2 and E3. The numbers in brackets are corresponding result ranges over fifty runs.

creature. When the CBG and MENL learn exploration in the simplest environment E1, and especially when they re-explore environment E1 after successive learning of E1 and E2, the exploration efficiency achieved is even better than that of the hand-crafted exploration.

Compared with fresh learning in E2, both successive learning in E2 after fresh learning in E1 and in E3 has shown clearly better performance. The final exploration efficiencies obtained by both successive learning are about 5 percentage points and 9 percentage points higher than that obtained by fresh learning. Moreover, successive learning has generated much fewer collisions and required much less evolutionary learning to learn successful exploration in E2 than fresh learning. The better performance of successive learning benefits from the knowledge transferred from the previously learned environments. Even when the creature

successively learned exploration in E1 and E2 explores the earlier environment E1 again, it can still conduct the exploration competently with only a few adjustments.

When the CBG and MENL learn exploration in the maze E3, the fresh learning and the successive learning after learning in E1 first and E2 next have achieved similar exploration efficiencies. Because the maze E3 is much more complicated than environment E1 and E2, both successive learning and fresh learning have encountered much difficulty in their exploration of E3. However, compared with fresh learning, successive learning in E3 has still exhibited a better overall learning performance since the average learning procedures and the average collisions required in total are fewer in successive learning. In addition, due to the exploration skills transferred from previous exploration learning, successive learning in E3 can start at a better exploration ability with higher exploration efficiency and lower learning and collision times.

When the CBG combined MENL learns behaviours continuously, the skills learned for executing the behaviours can be transferred not ony from simple to complex environments, but also from complex to simple environments. After the creature equipped with the CBG and MENL has learned exploration in the most complex environment E3, it can easily learn to explore the less complex environment E2 very well. With the exploration skills obtained from E3, the creature starts to explore E2 with an efficiency as high as 72%, which is similar to that achieved by the successive learning of E1 then E2. The successive learning from complex to simple environments has also exhibited both obviously fewer learning procedures and collisions, compared with other exploration learning in E2, such as fresh exploration and successive exploration from simple to complex environments. The cumulative learning cost of learning E3 then E2 is similar to that of learning E2 then E3.

Though the above experimental results have demonstrated that the CBG model combined with MENL can learn to adapt to a series of unknown environments, we also find that incremental learning in consecutive environments is often more expensive than direct learning of an environment. For instance, it costs more

to learn E1 then E2 then E3 than to learn just E3. Also, having learned in E3 alone it is then cheaper to learn E2 than to do E2 first. This suggests that it would be better for a virtual creature to learn the hard environment first if it knows which is the hard environment beforehand. However, because a virtual creature can move around continuously during its lifetime, it is always difficult to know what kind of environments it is going to meet. Actually the creatures can run across any unknown environments and situations with varied complexity at any time. If a creature learns each environment from scratch every time, it is obvious that it would take the creature a hugh amount of time and effort to suit all of the environments. But with the control of the CBG model, a virtual creature can have behaviours and abilities accumulated from environment to environment. The creature would be able to execute behaviours in new environments at a certain level, by using the knowedge learned from other environments. At the same time, the creature can learn to improve its behaviours continuously in new environments. Due to the knowledge transferred from other environments, incremental learning is usually much easier and more efficient than learning from scratch in a new environment alone. As learning is retained through its whole lifetime, a virtual creature can have accumulated and improved behaviours to suit a wide range of environments and situations. The property that learning of one environment implies knowledge of another environment helps the CBG model to be an adaptive and general behaviour control model suitable to virtual creatures.

4.4.4 Summary

In this section, we have conducted several exploration experiments in various unknown environments to test the adaptive learning ability of the CBG model combined with MENL. A virtual creature equipped with the CBG and MENL has been required to explore three unknown environments E1, E2 and E3 successively. The experimental results have shown that this creature has adaptively improved its exploration ability through evolutionary learning of multiagents, and learned how to explore all these environments successfully. Some comparison results have also demonstrated that the co-decision of multiagents is more robust and flexible

than the solo decisions made by the best agent; and multiagents with continuous evolutionary learning through their lifetime are more competent and adaptive than non-improving multiagents.

With the increasing complexity of the environments, the exploration learning of the virtual creature has experienced increased difficulty. However, compared with fresh learning in a complicated environment, successive learning that is achieved after learning in simple environments is always relatively easier and involves much fewer collisions and shorter learning time. In some cases (e.g. in E2), the learned exploration efficiency of successive learning is also better than that of fresh learning. It is obvious that the accumulated exploration experience learned from simple environments has helped the virtual creature to learn the exploration in successive complicated environments more easily and competently.

Hand-crafted exploration experiments have also been conducted in three unknown environments. In these experiments, a hand designed controller takes responsibility for choosing the best motor actions according to the information about the places in front of the creature. In each environment, the resulting exploration has traversed similar numbers of places to the exploration adaptively learned by the CBG with MENL. These results indicate that the adaptive learning of the CBG combined with MENL is almost as good as a hand-crafted controller for exploration.

4.5 Emergent Behaviours

In the above experiments, the virtual creature has exhibited many emergent behaviours. These behaviours are not imposed by pre-programmed mechanisms nor are they directly enforced by the fitness function. The behaviours are lifelike, efficient and robust. In this section, we use somewhat anthropomorphic language to describe the behaviours of the virtual creature, as Braitenberg does in [25].

1. Moving in a straight line

   One of the impressive emergent behaviours is the virtual creature's decision

to move in a straight line. As the above figures show, despite many possible choices, the creature prefers to go ahead. Moving straightforward is a common characteristic of walking animals. However, in many other works, this feature has been achieved by special design of the fitness function in which an extra term is used to favour straight movement by penalising turns [50, 137]. In contrast to these, the straight movement reported here has emerged out of our design.

2. Avoiding stagnation

When surrounded by obstacles, the moving behaviour may become incompatible with the avoidance behaviours. This problem is referred to as *stagnation* [5]. A variety of strategies have been suggested for overcoming this weakness, mainly using additional constraints to force the creature to move [4, 85]. However, through evolutionary learning under a relaxed fitness function with fewer constraints, our creature can autonomously learn to find available paths to free itself from the predicament. Figure 4.7 shows an example of avoiding stagnation, in which the virtual creature learns to avoid the Γ-shaped obstacle via evolution strategies.

3. Active direction selection

Sometimes the creature may not step out from an impasse immediately, especially when the visual field is almost full of obstacles. The exploration trajectory shown in Figure 4.7 is a good example. When the creature starts from the bottom left corner, it is surrounded by walls and obstacles. In this situation, the simulated creature attempts to adjust its visual direction actively, instead of moving impetuously. A final decision is made after viewing more situation details from varied directions. It seems that this creature collects more environmental information and considers it carefully. Such a careful decision is often a unique or better choice in such a situation.

The above behaviours have emerged out of the design. By walking in a straight line and selecting actions after careful "consideration", the creature can move

more efficiently. By avoiding stagnation actively, the creature's behaviours are more robust to its environments. Moreover, the emergent behaviours look quite natural and believable. They can often be found when living animals move in their environments. These behaviours can help a virtual creature look more like a real living creature and hence enhance its believability to the human user.

The emergent behaviours are not independent of each other. They can happen in parallel and result in more sophisticated motions, such as walking in a narrow aisle without collision, following a wall, stepping out from a corner, moving around a big obstacle to avoid it, and so on. These advantages may benefit from the multi-agent engine which is evolved under a relaxed fitness function. While a high-ranked agent obtains an initiative to control the movement, it is quite possible that this agent will win again in the next competition due to its successful experience. Therefore, a good movement can be repeated and some natural behaviours (e.g., moving ahead) can be generated. When the virtual creature encounters problems and current agents cannot take positive roles, new candidates are sought with the assistance of evolutionary learning. Unconstrained by many specific requirements, the evolutionary search proceeds in a broad space so that there is a high possibility of finding feasible solutions to avoid a series of predicaments.

4.6 Conclusion

This chapter presents a learning algorithm, Multi-agent based Evolutionary artificial Neural network with Lifetime learning (MENL), which can learn to make correct decisions of motor action selection based on sensory feedback and past experience. The MENL has a number of notable merits:

- There are a batch of multiagents maintained in MENL, which cooperate and compete with each other for the decisions of action selection. This co-decision mechanism enhances the adaptation and robustness of behaviours.
- The evolutionary learning of the multiagents is kept through the lifetime of

MENL. In consequence MENL can learn to improve its ability continuously, and is able to cope with more and more situations in its lifetime.

- The fitness function of MENL is a relaxed design, so the evolutionary search may proceed in a broad space with few constraints. The fitness function is also designed for more than one behaviour, since it takes account of both general sensory feedback on executed actions and specific behavioural objectives.

When MENL is embedded in the Strategy module of the CBG model to choose motor actions, its learning and decision-making become meaningful: the CBG model carries out motor actions chosen by MENL in practice and reports execution results of these actions through continuous sensory feedback. Any mismatch of intended actions and actual action execution results will urge MENL to use evolution strategies to revise unqualified agents that have generated results unexpected and/or harmful to the achievement of a behaviour. On the other hand, the successful execution of intended actions will be memorised implicitly in multiagents (i.e., ANN individuals) for future use. By getting rid of the unfit agents and retaining the essential, MENL can improve its ability continuously in response to changing situations.

A virtual creature is equipped with a CBG model combined with MENL to execute exploration in several unknown environments. Experimental results have shown that the virtual creature has adapted to various unknown environments and explored them successfully. The exploration ability of the creature is obviously increased through trial and error and this ability is accumulative from environment to environment. Because the creature learns behaviours from scratch and based on the information about the currently perceived environment, the learned exploration ability is general and robust, and not specific to any particular environment or object. During exploration, there is always more than one agent that contributes different but useful ideas to the exploration in various situations. The whole exploration is implemented by the competition and cooperation of multiagents. In experiments, evolutionary multiagents with lifetime learning have also exhibited much more robust and adaptive ability than non-improving

multiagents. It is the lifetime learning that helps the creature to enrich its exploration experience constantly. As a result, the creature has achieved exploration at a performance level similar to that of hand-crafted exploration which is based on information about the places immediately next to the creature.

Through the combination with Multi-agent based Evolutionary Artificial Neural Network with Lifetime Learning (MENL), the CBG model has produced adaptive exploration in many environments. The resulting exploration is flexible to various situations, able to improve from mistakes, and proactive for a specific behavioural motivation. This kind of exploration, one of the space occupying behaviours, is fundamental to many living creatures.

In the next chapter we will introduce the multifunctional learning of the CBG combined with MENL and show how they are used to learn several space occupying behaviours together.

Chapter 5

Learning Multiple Behaviours: Combining the CBG Model with the MENL Learning Algorithm

In this chapter, we present multifunctional learning of the CBG model combined with the MENL learning algorithm. In the next section, we give a brief review of recent work on knowledge transfer in artificial neural networks and relate our work to this. Some biological findings on multifunctional neural networks for natural behaviour control are explained in Section 5.2, which provide the main inspiration and basis for our work. In Section 5.3, we introduce the multifunctional learning of the CBG and MENL for producing multiple behaviours. Section 5.4 describes the experiments we have conducted when the CBG with MENL is applied to learn multiple space occupying behaviours. The last section concludes with a summary of this chapter.

5.1 Related Work

Recently there has been increasing interest in transferring knowledge learned by artificial neural networks across related tasks. This is initiated by the realisation that complex environments will require learning to perform multiple tasks [146]

and the learning can be greatly saved or simplified if the tasks to be learned can share what they learn [33, 125]. In this section, we review some typical work in this area and put our work in the context.

Pratt et al. have studied the sequential transfer of learned knowledge between artificial neural nets [124, 125]. In their previous study, Pratt et al. presented somewhat surprisingly better effects of pre-setting neural network weights on subsequent learning, compared with untrained weights [124]. They demonstrated that the relative magnitudes of the preset weights were important for retaining the locations of pre-trained hyperplanes. Pratt also proposed a Discriminability-Based Transfer (DBT) algorithm to estimate the utility of hyperplanes defined by source weights in the target network and to rescale transferred weight magnitudes accordingly [125]. The DBT algorithm used the target training data and weights from the source network, along with two coefficients to calculate a modified set of weights for initialising training on the target task. Several empirical results on speech recognition, disease diagnosis, pattern recognition, and chess problems proved that target networks initialised via DBT learned significantly faster than networks initialised randomly. Similar results have also been reported by Sharkey and Sharkey, which have shown that what was learned for one task could be used as a good bias for other tasks [144].

In addition to speeding up learning, improved generalisation on sequences of learned tasks has been studied as well [33, 156, 157]. Thrun et al. suggested study of robot learning problems not in isolation, but in the context of the multitude of learning problems that a robot would face over its entire lifetime [156, 157]. They proposed an Explanation-Based Neural Network (EBNN) to generate target values by explaining and analysing each observed training example of the target function in terms of the domain theory acquired in previous learning tasks [157]. Action models that captured important domain knowledge independently of the particular control learning problem at hand were used to bias learning of the control function of the current problem. In an experiment concerned with grabbing a cup, for instance, EBNN learned the evaluation function based on pretrained action models, and discovered the correlation of the distance of the

cup and the success of the *grab* action [157]. Even when the observed training examples were very few, the evaluation function learned by EBNN still had a more correct shape than plain learning with no transferred knowledge from action models. As a consequence, EBNN was able to replace real-world experiments efficiently and support accurate learning by previously learned bias.

Caruana [33] has proposed multitask learning (MTL), an inductive transfer mechanism that improves task generalisation by taking advantage of domain information contained in the training signals of related tasks. Specifically, multitask learning could be achieved by learning several related tasks in parallel while using a shared representation (e.g., a shared hidden layer in a backpropagation ANN). Therefore, what was learned for each task may help other tasks to be learned better. Sometimes, better performances of main tasks were obtained by learning extra features of other tasks. Caruana has implemented multitask learning in several different mechanisms, such as backpropagation ANNs, K-nearest neighbour and decision trees [33]. The ability to achieve multitask learning with different inductive methods pointed to the generality of its basic idea. Nevertheless, the principal goal of MTL was to improve the performance of learning on main tasks by using extra outputs of related tasks. For better performance on the main tasks, the learner allowed performance on the extra tasks to degrade or could even ignore the extra tasks.

Ijspeert, Hallam, and Willshaw have studied how neural swimming controllers for a lamprey could be adapted for controlling both the swimming and the walking of a salamander-like animat [74]. Initially, they used genetic algorithms to evolve over several stages the swimming controllers of a simulated lamprey, which had similar neural configurations to a biological connectionist model of the Central Pattern Generators (CPGs) controlling the swimming of a real lamprey [75]. The evolved controllers produced neural connections and control efficiency close to those observed in the real lamprey. Ijspeert et al. then extended the swimming controllers of the simulated lamprey to control the swimming and walking of a simulated salamander. As little was known about the neuronal circuits underlying the locomotion of salamanders, the walking controllers of the simulated

salamander were evolved based on the structure of two interconnected oscillatory networks of the swimming controllers. All neurones of the swimming and walking controllers received simple excitatory signals from the "brain stem" through different pathways. Therefore, swimming or walking patterns were generated depending on which controllers were excited, and the speed of locomotion was proportional to the level of excitation. The successful development of the walking controllers based on the mechanisms transferred from the swimming controllers suggest a reasonable conjecture that the oscillators for the limbs have followed a corresponding specialisation from the trunk segmental networks through natural evolution.

These studies on knowledge transfer in artificial neural networks have demonstrated that learning of a new task can be achieved based on previous learning of other tasks, and the subsequent learning of new tasks may have accelerated the learning speed or improved the generalisation ability. However, the knowledge transfer usually happened between two different mechanisms in these studies, and the transfer was only from one task to another rather than mutual knowledge transfer between different tasks. Also, it is not clear whether the mechanisms (ANNs) using transferred knowledge to learn new tasks can integrate knowledge of different tasks and implement both previous and new tasks well. Although Caruana used the same neural network to learn multitasks, the purpose of the multitask learning was to improve the performance of the main tasks, not all of the multiple tasks. In addition, there is more than one output of the neural network, each of which is for a particular task. Differing from the above work of knowledge transfer, our study on multifunctional learning uses the same mechanisms (MENL and its ANNs) to learn varied action selection policies equally well for more than one behaviour. The artificial neural networks of MENL are evolved as a whole to learn for multiple behaviours. In consequence the common knowledge shared in executing these behaviours is transferred from one to another, and the different properties of these behaviours are naturally integrated together via evolutionary learning. The resultant MENL can therefore execute

multiple action selection policies and the total time spending on learning these policies together is much lesser than that on learning these policies individually. During its multifunctional learning, MENL works just like those biological multifunctional neural networks producing different output for implementing different behaviours. The multifunctional neural networks in biological behaviour control are introduced in the next section.

5.2 Biological Multifunctional Neural Networks

It is well known that the brain can perform numerous functions although its anatomical structure and size are finite. This powerful feature of the brain may be contributed to shared knowledge and resources across related behaviours and multifunctional neural circuits that can be modulated for the purpose of performing more than one specific function [43, 56, 122, 142]. The multifunctionality of these neural circuits doesn't mean that an olfactory system can be used as an auditory system. But, within the confines of the anatomical substrate (the anatomical organisation), the functional organisation of multifunctional neural networks appears to be under dynamic control, changing in accordance with the expression and modulation of the constituent cellular, synaptic, and network building blocks [56].

The concept of *multifunctional neural networks* was proposed and elaborated by Getting and Dekin [55] to explain the control of swimming and defensive withdrawal in the nudibranch mollusc *Tritonia*. In their study, Getting and Dekin found that afferent depolarisation of one set of premotor interneurons reconfigures a pattern-generating circuit from a state in which it generates a pattern for withdrawal to a state in which it generated the pattern for swimming [55]. Thereafter, numerous recordings from neurones have noted that many biological neural networks participate in a series of different but functionally related behaviours, including walking, scratching and posture in cats [54]; pyloric and gastric mill rhythms in the stomatogastric system of crabs [172]; jumping and kicking in the locust [66]; rhythmic hatching and stepping movements of legs of chicks [17]; dif-

ferent forms of scratching in the turtle [113]; and so on. The neural networks controlling one behaviour may share a portion or even all of their neurones with networks controlling other behaviours.

At present, the actual mechanism for configuring multifunctional neural circuits is not yet clear. However, it is known that, at least in the case of central pattern generators (CPGs) (neural networks underlying rhythmic movements), the nature of afferent, central, and neuromodulatory signals are the main sources of input that could act to build varied neural circuits [122, 142]. While the brain adjusts these inputs intentionally, CPGs can change or even discontinue their behaviour patterns accordingly, and elicit interactions with other functionally diverse circuits in many different ways. As a consequence, the neural networks responsible for basic rhythmic movements are not fixed, but flexible and multifunctional systems capable of producing different motor output patterns.

5.3 Multifunctional Learning of the CBG Combined with MENL

After giving the virtual creature exploration ability, we want the virtual creature to also be able to produce some other space occupying behaviours, such as reaching goal locations, wandering randomly, and so on. Hence, the CBG model combined with MENL needs to select suitable motor actions and execute these actions for carrying out different space occupying behaviours. All these behaviours, including exploration, goal reaching, and wandering, share the same sensors and motors, motor programs, motor actions, and even part of the motor action selection policies, as we introduced in Section 3.5.2. The only difference among these behaviours is that, guided by different behavioural motivations, the exact motor action selected at a particular time may be different for each behaviour. If we use every single MENL learning algorithm to learn the correct motor action selection policy for every behaviour, we would have a collection of MENLs in the CBG model. While the number of behaviours to be implemented increases, it is questionable if it is economical and practical for the CBG model to hold many

MENLs and corresponding multiagents (artificial neural networks) with similar functionality within a limited structure and space. In order to sensibly implement multiple behaviours in the same behaviour control model, we hope MENL and its ANNs can learn multiple motor action selection policies for multiple behaviours, by taking advantage of common properties shared in these behaviours.

In the last section, we introduced the biological multifunctional neural networks in the brain, which can participate in or generate more than one behaviour [43, 56, 122, 142]. The generated behaviours usually involve either the same muscle groups or functionally related muscle groups, but are driven for diverse objectives. Inspired by biological multifunctional neural networks, we consider constructing multifunctional artificial neural networks in MENL and the CBG model. As the CBG model provides basic mechanisms (many shared resources at various layers) for potential multifunctionality, we hope the MENL learning algorithm can implement this potential, by choosing correct motor actions for multiple behaviours in the same artificial neural networks and by utilising the common knowledge shared between these behaviours. If the CBG model with MENL can adaptively learn to implement functionally related behaviours according to different behavioural objectives, not only would the required mechanisms (e.g., neural networks) be much reduced, but the common knowledge and resources across these behaviours would also have the opportunity of being used efficiently.

In Chapter 4, we presented the adaptive learning of exploration achieved by the CBG and MENL. The multiagents maintained in MENL were driven by the behavioural objective in the fitness function to learn the proper action selection policy for exploration in various unknown environments. In this chapter, we will change the behavioural objective in the fitness function so that MENL and its multiagents can learn to carry out some other behaviours. Moreover, we hope MENL agents can appropriately integrate the knowledge learned for different behaviours so MENL can be multifunctional and implement different behaviours in the same ANNs. In addition to the behavioural objectives, the item of sensory feedback in the fitness function provides the learning of MENL with some general

knowledge common to behaviours of the same kind. The common knowledge across functionally related behaviours ought to help MENL to learn multiple behaviours more competently and effectively.

In the following section, we will test the potential multifunctionality held in the CBG and MENL by doing some experiments. In particular, we will command the CBG combined with MENL to learn to produce several space occupying behaviours together in some unknown environments, including exploration, goal reaching, and wandering. The downstreaming central signals from the Motivation module of the CBG model are the main impetus to drive the artificial neural networks of MENL to work for different behaviours. The experiments will test whether the CBG with MENL can learn to perform more than one behaviour together, and if it can, how good its multifunctional learning performance is.

5.4 Experiments

In this section, the abstract bot virtual creature embedded with a CBG and MENL is again used but this time to carry out multifunctional learning. Several experiments on learning multiple space occupying behaviours have been designed, with differently complex environments and different numbers of behaviours to learn. In the first experiment, the virtual creature is required to learn two kinds of space occupying behaviours (exploration and goal reaching) jointly in a simple environment E1. The creature is also used to learn these two space occupying behaviours together in a more complicated environment E2. In the last experiment, the creature tries to learn three kinds of space occupying behaviours (i.e., exploration, goal reaching, and wandering) together in E2. Some interesting trajectories generated in the experiments are also presented, which show the robust and continuously improved behaviours of the virtual creature.

5.4.1 Multifunctional Learning and Independent Learning

When more than one different behaviour is learned in a common CBG model, especially in the same artificial neural networks, the successful learning of one

behaviour usually cannot guarantee the integrity of other behaviours learned before. In fact, due to the disparity between these behaviours, learning of an extra behaviour may bring some negative effects to the other learned behaviours. In Section 3.5.2, we exemplified that the actual motor actions selected may be different for generating different space occupying behaviours in some cases even when all the other conditions are the same. So, if artificial neural networks that have learned the successful action selection policy for exploration are trained to learn a new action selection policy for goal reaching, it is possible that the neural networks are re-stored with new learned knowledge for goal reaching and the collection of the previously learned knowledge for exploration is diminished. In order to execute both exploration and goal reaching behaviours successfully, the artificial neural networks need to re-learn exploration and maybe goal reaching many times so as to remember and integrate different knowledge for different behaviours correctly. Therefore, in multifunctional learning of the CBG and MENL, learning of multiple behaviours will be conducted alternately until every behavioural learning is executed smoothly. Figure 5.1 examplifies the experimental procedures of learning three behaviours in multifunctional learning.

Although the learning of different behaviours may adversely affect each other, the learning processes may have mutual benefits too because functionally related behaviours have much shared common knowledge, as explained in Section 3.5.2. If the common knowledge is obtained by learning of one behaviour and is applied to learning of other behaviours properly, the learning of other behaviours will become easier and more efficient. To test this claim, independent learning of three space occupying behaviours has been conducted. In independent learning, every space occupying behaviour is learned on a randomly initialised CBG model and MENL, and a behaviour is considered to have been learned after one hundred consecutive trajectories are conducted successfully. Learning cost and learning results of independent learning will be compared with those of multifunctional learning.

In both multifunctional and independent learning, learning of wandering requires the virtual creature to move from a random starting point and keep mov-

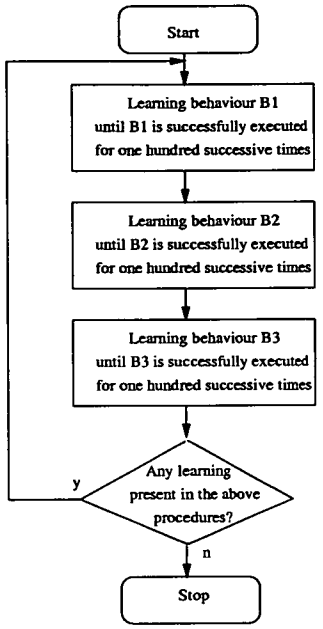


Figure 5.1: Experimental procedures of learning multiple behaviours in multifunctional learning

ing for a limited number of steps in each trajectory. (The limitation is 240 steps in environment E1 and 650 steps in E2.) Whenever the creature makes a collision, evolutionary strategies will be immediately triggered to evolve new qualified agents in MENL. The learning of wandering is completed (or temporarily finished in multifunctional learning) if the creature conducts one hundred successive trajectories successfully. Evolutionary learning times and collision times taken for the learning are performance indices to evaluate wandering learning ability.

In exploration learning, the creature needs to explore the environment as far as possible in the same limited number of steps as in wandering learning. All the starting points of exploration are randomly chosen. The exploration learning is finished when the creature executes one hundred successive trajectories successfully. When independent exploration or an episode of exploration in multifunctional learning has been learned, in addition to learning times and collision times, another performance index used to test the learning ability is the final exploration efficiency achieved by successful exploration learning.

In goal reaching learning, the creature, starting from arbitrary initial places and orientations, is required to reach randomly selected goal destinations. The creature should find an available route to the goal in the limited number of steps, the same as that of wandering and exploration learning. Likewise, goal reaching learning finishes when one hundred successive goal destinations are reached successfully. A goal reaching trajectory is deemed as a "success" if the goal is achieved within the limited number of steps; otherwise the conduction of goal reaching is a "failure". In addition to evolutionary learning times and collision times, another two indices are adopted to estimate the performance of goal reaching learning. One index is goal reaching efficiency, that is, the steps involving movement as a percentage of all the steps the creature makes. Those steps causing collisions or fixation at a place are thought to impair the navigation. The other performance index is the success rate of goal reaching, which indicates whether the creature has reached the goal destination in each trajectory. Because every goal reaching experiment is conducted for fifty runs, the success rate is actually the probability of conducting a successful trajectory in fifty runs. One experimental run of goal reaching learning is finished when the creature conducts one hundred successful goal reaching trajectories successively.

The settings of the CBG and MENL are those described in Section 3.7 and 4.3 for both multifunctional and independent learning. The fitness functions adopted for exploration, goal reaching, and wandering are equations 4.5, 4.6, and 4.7 respectively.

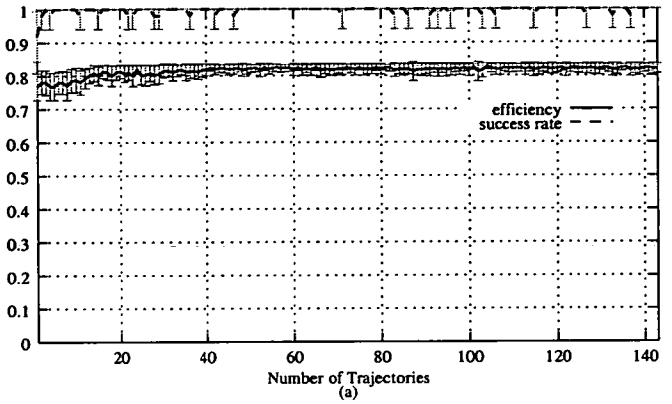
It may be worth reminding the reader that the key idea of the MENL learning algorithm is to find a general solution that can be applied to a wide range of situations. Sometimes, this solution may not be optimal at all. Similarly, in multifunctional learning, the main contribution of MENL is to seek a general solution that can be applied to multiple behaviours in a wide range of situations, rather than to optimise every behaviour in some specific situations. However, the investigation of a better compromise between generalisation and optimisation is part of our future work.

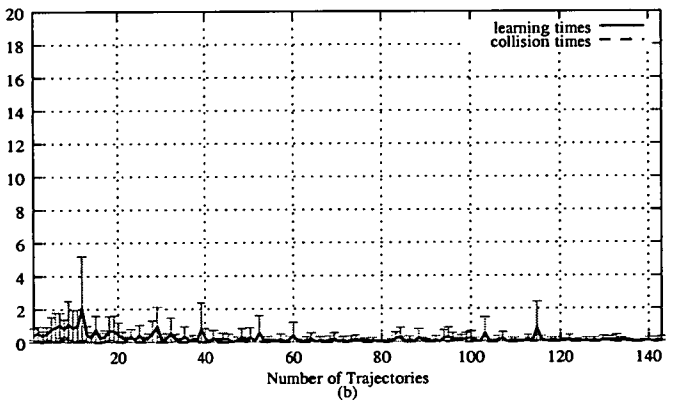
5.4.2 Learning Two Space Occupying Behaviours Together in the Simple Environment E1

In this experiment, the virtual creature with an initialised CBG and MENL attempts to learn exploration and goal reaching together in a simple environment E1. The main purpose of this experiment is to check if the CBG with MENL can learn more than one behaviour, and how the learning of one behaviour affects another. The virtual creature that has already obtained exploration skills through fresh exploration learning (i.e., independent exploration learning) in E1 is firstly trained to learn a second behaviour of goal reaching. Further exploration is then executed for an estimation of the influence of the extra learning of goal reaching on the previously learned exploration ability. Even more goal reaching and exploration learning are executed one after another so the creature can learn both behaviours appropriately. Finally, a comparison between the performance of multifunctional learning and independent learning is presented.

5.4.2.1 Subsequent Learning of Goal Reaching After Fresh Exploration

The first part of the experiment is the additional learning of goal reaching in environment E1. The virtual creature has just learned exploration in E1 and hence possesses certain exploration and obstacle avoidance skills (experimental information of fresh exploration learning in E1 is presented in Section 4.4.2.1, so we won't repeat it here). In subsequent goal reaching learning, we want the creature to reach arbitrary goal destinations from random initial positions and orientations in 240 steps until it executes successful goal reaching one hundred successive times. Every creature generated from the fifty runs of fresh exploration learning in E1 is used to conduct subsequent goal reaching learning. The goal reaching efficiency and success rate of reaching a goal destination in each trajectory averaged over fifty runs are shown in Figure 5.2 (a). Figure 5.2 (b) shows the number of evolutionary learning procedures and the number of collisions the creature makes in each trajectory, also averaged over fifty runs. Figure 5.2 (c) lists the average total learning times and collision times spent in fifty runs of the subsequent goal reaching learning, and the average goal reaching efficiency and





Total learning times	Total collision times	Final efficiency	Final success rate
31.9	2.2	82.5%	100%
(0-68.9)	(0-5.0)	(81.6%-83.4%)	(100%-100%)

(c)

Figure 5.2: Subsequent goal reaching learning after fresh exploration in E1 (averaged over fifty runs). Each creature of the fifty runs of fresh exploration in E1 is applied to reach various randomly selected goal destinations in no more than 240 steps until the creature has reached one hundred successive goal destinations successfully. The goal reaching efficiency and success rate of each trajectory averaged over fifty runs are recorded in Figure (a). The average learning times and collision times spent in each trajectory over fifty runs are shown in Figure (b). The last 100 successive and successful trajectories are not shown in Figure (a) and Figure (b). Figure (c) lists the total numbers of learning procedures and collisions required for the subsequent goal reaching learning, and the goal reaching efficiency and success rate obtained at the end of the learning, all averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

success rate achieved when the learning finishes.

The results in Figure 5.2 have shown that the virtual creature has learned the new behaviour of goal reaching successfully in subsequent goal reaching learning. In addition, the creature implemented this learning with ease. There are very few collisions made in the whole experiment: the average number of total collisions is only 2.2 in fifty runs . The learning times are also kept very small. The number of learning procedures is no more than 2 in each trajectory, and it gradually decreases to be lower than 0.1 in the last twenty trajectories. The average number of total learning procedures is 31.9 in fifty runs.

The goal reaching efficiency of subsequent goal reaching learning starts from a very good point of 77%. It is slowly raised to 82.5% when the learning finishes. This improvement indicates the virtual creature spends more time on sensible movement towards the goal destination, and there is only 17.5 percent of its movements wasted on hesitation, direction changing, etc, which have no direct contribution to the navigation. The last noticeable result presented in the subsequent learning is the virtual creature's very high success rate in reaching goal destinations. Even at the beginning of the experiment, the probability of reaching a random goal destination in fifty runs is as high as 92%. When the experiment finishes, the creature has learned how to locate various goal destinations in unknown environment E1, and reached all destinations in the limited number of steps without problem. The success rate of goal reaching is therefore 100% at the end of the subsequent goal reaching learning.

Through deliberated adjustment of its ANN structures that were first set for exploration, the virtual creature has learned a second behaviour of goal reaching successfully.

Independent Goal Reaching Learning

From the good performance of the subsequent goal reaching learning after fresh exploration, we conjecture that the previously learned exploration may have some good effects on the later learned goal reaching. For a demonstration of this conjecture, we use a fresh virtual creature with an initialised CBG and MENL to

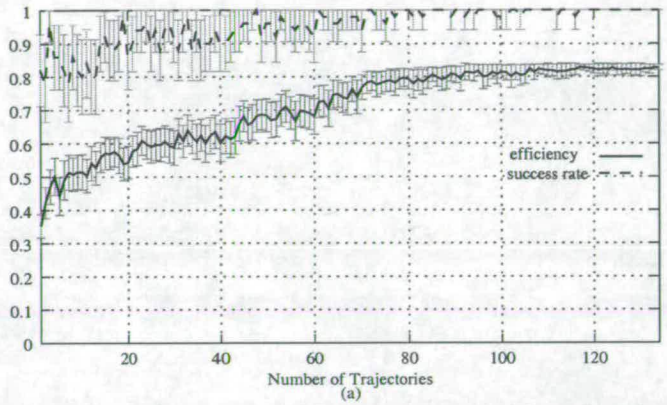
conduct independent goal reaching learning in environment E1. Similarly, the creature should manage to reach its goal destinations starting from arbitrary places and orientations in 240 steps while keeping its obstacle avoidance competence. This learning is repeated fifty times, by using different initialisation of the creature in each learning episode. The averaged learning results over fifty runs are shown in Figure 5.3.

As expected, the fresh creature has acquired goal reaching ability through adaptive learning of the CBG and MENL. However, compared with the subsequent learning, the independent goal reaching learning has presented worse learning performance. There is intensive evolutionary learning involved in the independent learning. The total learning sessions are as many as 364.2, an average over fifty runs. This number is almost 11 times more than that required by the subsequent learning of goal reaching. The initial goal reaching ability the fresh creature displays at the beginning of the independent learning is also poor. The initial efficiency is only 35.5% and the initial success rate is 80%. They are 35 points and 12 points lower than those of the subsequent goal reaching learning respectively. Both subsequent and independent goal reaching learning have achieved similar navigation efficiency (82.5% and 82.9% respectively) and the same success rate (100%) in the end.

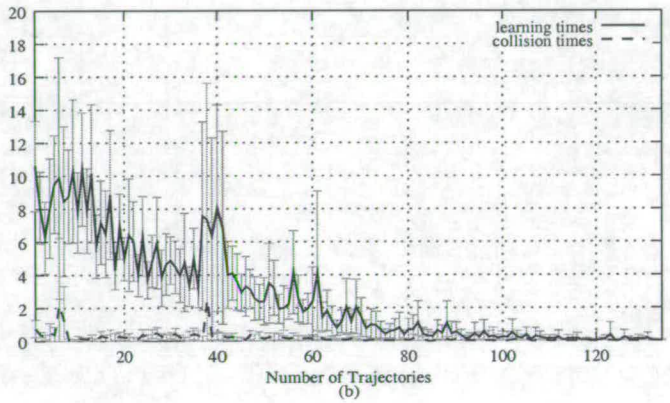
Because the subsequent goal reaching learning starts from the point which is defined by the previous exploration learning, the learning of the subsequent goal reaching is much easier and smoother than the independent learning starting from random. This result suggests that some useful knowledge contained in exploration learning has been utilised or transferred to goal reaching learning. The good performance of the subsequent goal reaching learning also supports Pratt's inference that an appropriate pre-setting for neural network weights plays a positive role on its subsequent learning [124].

5.4.2.2 Further Exploration

After extra training in goal reaching, the creature needs to explore E1 again, for a test of whether it still remembers the previously learned exploration ability. The



(a)



(b)

Total learning times	Total collision times	Final efficiency	Final success rate
364.2 (291.8-436.6)	23.4 (13.6-33.2)	82.9% (82.2%-83.6%)	100% (100%-100%)

(c)

Figure 5.3: Independent goal reaching learning in E1 (averaged over fifty runs). In each run, a fresh virtual creature equipped with a randomly initialised CBG and MENL is asked to reach various random goal destinations from arbitrary places and orientations in no more than 240 steps. The learning is finished when one hundred successive goal destinations are reached successfully. The goal reaching efficiency and success rate of each trajectory averaged over fifty runs are recorded in Figure (a). The average learning times and collision times spent in each trajectory over fifty runs are shown in Figure (b). The last 100 successive and successful trajectories are not shown in Figure (a) and Figure (b). Figure (c) lists the total numbers of learning procedures and collisions required for independent goal reaching learning, and the final goal reaching efficiency and success rate obtained by the learning, all averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

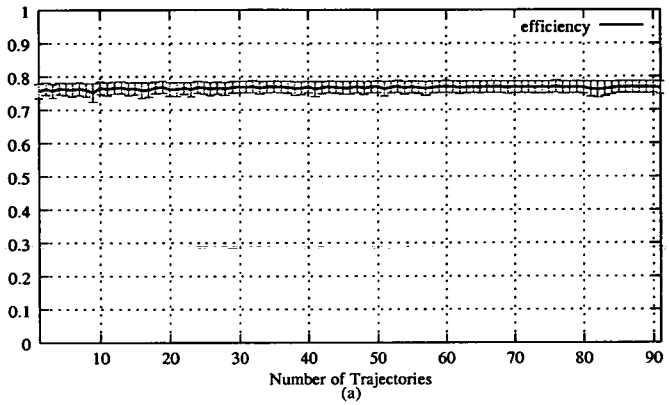
creature should continuously explore environment E1 starting from various random points in 240 steps, until one hundred successive and successful exploration trajectories have been executed. Every virtual creature of the fifty runs of subsequent goal reaching learning conducts further exploration learning. Figure 5.4 shows the results of the second exploration learning in environment E1, averaged over fifty runs.

The experimental results have shown that there are some collisions and evolutionary learning during further exploration, however, both collisions and learning procedures are very few. For instance, the total collisions made in the whole exploration are only 1.7, and the total learning times are only 16.9 (averaged over fifty runs). At the same time, the exploration efficiency of each trajectory is maintained at a relatively stable state of 76.7%. This efficiency is similar to that obtained by fresh exploration learning, which is 75.7%. After exploring the environment E1 even further (a maximum of ninety one trajectories in fifty runs), the virtual creature has properly recollected its early learned exploration ability.

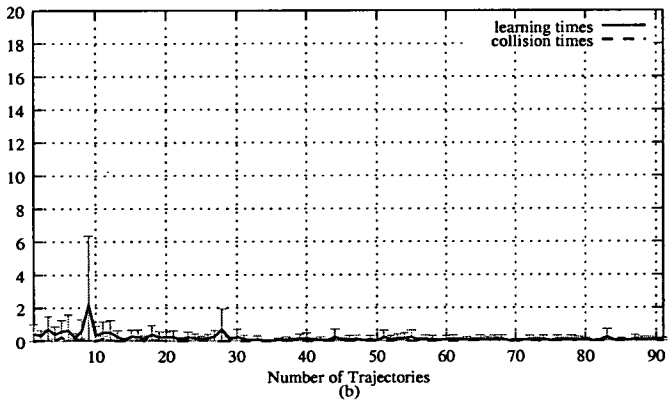
5.4.2.3 Further Goal Reaching

After the successful learning of further exploration, the creature is put in environment E1 again but to execute further goal reaching. This is for an inspection of the further exploration's influence upon goal reaching. The creature starts from random places and orientations and tries to reach various goal destinations in no more than 240 steps. Further goal reaching continues until the creature reaches one hundred successive goal destinations successfully. Similarly, every creature of the previous fifty runs of further exploration is used to conduct a further goal reaching experiment. The experimental results averaged over fifty runs are shown in Figure 5.5.

From the experimental results, we can see that there are only very occasional collisions and unreached goal destinations involved in further goal reaching experiments. The average number of total collisions made over fifty runs is less than one, and the average number of total learning procedures is only 3.8. Meanwhile, the virtual creature can reach most goal destinations in fifty runs. It is



(a)

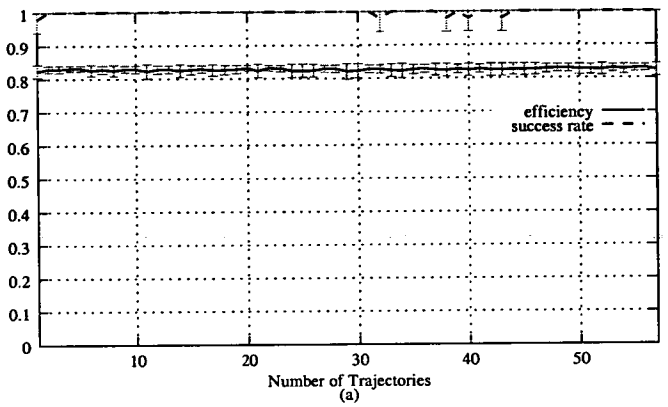


(b)

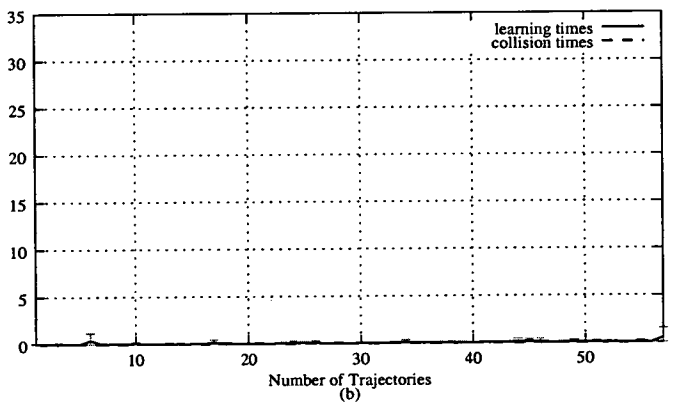
Total learning times	Total collision times	Final efficiency
16.9 (0-38)	1.7 (0-3.8)	76.7% (74.9%-78.5%)

(c)

Figure 5.4: Further exploration learning in E1 (averaged over fifty runs). The virtual creature which learned subsequent goal reaching ability is asked to conduct another exploration experiment to recollect its previously learned exploration ability. Every virtual creature resulting from the fifty runs of subsequent goal reaching learning conducts continuous exploration trajectories, until one hundred successive and successful exploration trajectories have been executed. The exploration efficiency reached by each trajectory is shown in Figure (a). The learning times and collision times made in each trajectory are shown in Figure (b). Both figures show results averaged over fifty runs and do not include the last 100 successive and successful trajectories. Figure (c) is a summary of the total learning times and collision times spent in further exploration learning and the exploration efficiency finally achieved by the learning, averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.



(a)



(b)

Total learning times	Total collision times	Final efficiency	Final success rate
3.8 (0-11.3)	0.2 (0-0.6)	84.0% (83.2%-84.8%)	100% (100%-100%)

(c)

Figure 5.5: Further goal reaching learning in E1 (averaged over fifty runs). Every creature of the previous fifty runs of further exploration in E1 is applied to reach various randomly selected goal destinations in no more than 240 steps until the creature has reached one hundred successive goal destinations successfully. The goal reaching efficiency and success rate of each trajectory averaged over fifty runs are recorded in Figure (a). The average learning times and collision times spent in each trajectory over fifty runs are shown in Figure (b). The last 100 successive and successful trajectories are not shown in Figure (a) and Figure (b). Figure (c) lists the total numbers of learning procedures and collisions required for further goal reaching learning, and the final goal reaching efficiency and success rate obtained by the learning, all averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

only in a few cases that the creature could not find the goals. The efficiency of the goal reaching is also quite stable: it is about 84% throughout the experiment. Despite the further exploration conducted beforehand, the virtual creature maintains most of its learned goal reaching ability.

5.4.2.4 Last Exploration and Goal Reaching

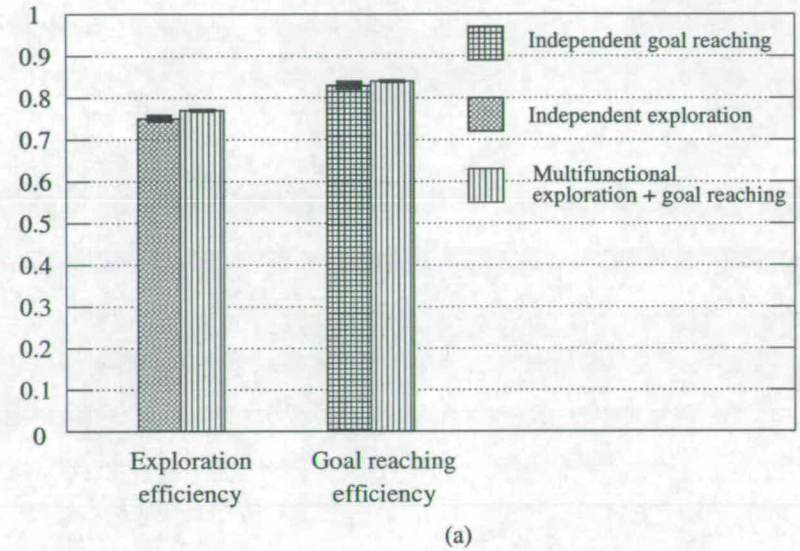
In the above further exploration and goal reaching, there is still some evolutionary learning occurring in both experiments. For real multifunctionality, the creature should be able to switch from one behaviour to another smoothly without learning. In order to test if the creature has already learned two space occupying behaviours completely, or if it still needs even further learning, a third set of exploration and goal reaching experiments are conducted in environment E1. The fifty creatures of the second goal reaching learning should first conduct continuous exploration trajectories until one hundred successive trajectories are executed without collision and learning, and then continuously locate various goal destinations from arbitrary places until one hundred successive destinations are reached without problem.

On the third repetition, all the creatures in fifty runs have successfully executed both exploration and goal reaching experiments successfully with no involvement of any collision or evolutionary learning. These creatures can now conduct either exploration or goal reaching well. Meanwhile, the average exploration efficiency is about 77.1%, and the average goal reaching efficiency is about 84.1%, while the creature reaches all goal destinations successfully.

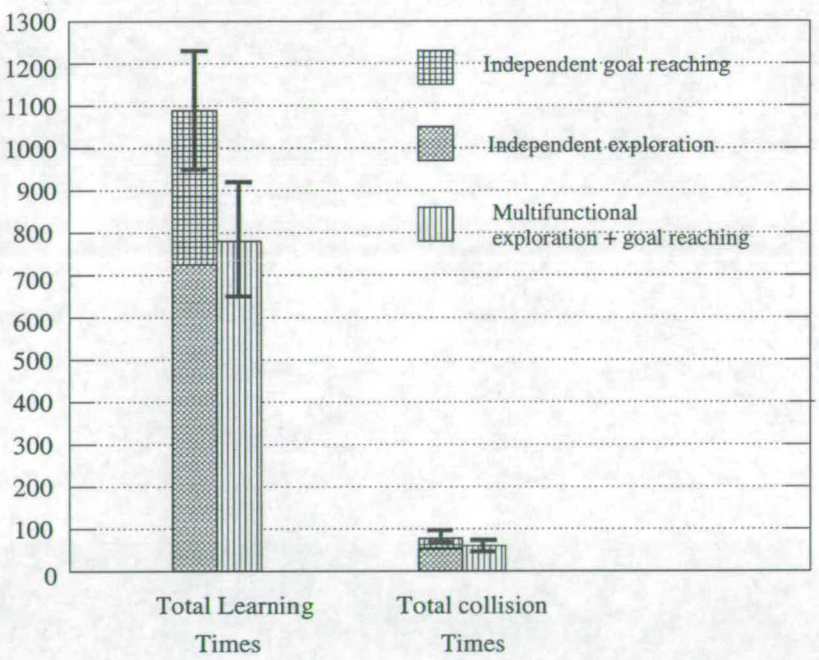
After three sessions of multifunctional learning procedures, the virtual creature at last has the multifunctionality of both exploration and goal reaching. While the multifunctional learning proceeded, the creature switched between these two behaviours more and more smoothly. The successful learning of exploration and goal reaching together shows that the CBG model combined with the MENL learning algorithm has the capability of learning more than one behaviour.

5.4.2.5 Comparison of Multifunctional Learning with Independent Learning

Figure 5.6 presents the comparison of multifunctional learning of exploration and goal reaching in environment E1 and the corresponding independent learning of exploration and goal reaching. As shown in Figure 5.6 (a), multifunctional learning and independent learning have achieved similar efficiencies in both exploration and goal reaching. But the total collision numbers made in the multifunctional learning is on average 20 less than the sum of the collisions made in every single behaviour learning, shown by Figure 5.6 (b). The clear superiority of multifunctional learning over independent learning is reflected in much obviously saved learning. Where independent exploration learning has spent 726.8 learning procedures learning single exploration ability and independent goal reaching has spent 364.2 learning procedures learning single goal reaching ability, multifunctional learning of exploration and goal reaching has only required 779.4 evolutionary learning procedures to learn both behaviours properly. After learning exploration independently, the same artificial neural networks used for multifunctional learning have only used 52.6 evolutionary learning procedures to adjust its functionality to comprise a new behaviour of goal reaching and to mediate different knowledge learned for different behaviours. Therefore, about 312 sessions of evolutionary learning have been saved in total in multifunctional learning, which is almost one third of the sum of the total learning times spent in independent exploration and goal reaching learning. As independent behaviour learning utilises two copies of multiagents (artificial neural networks) to learn action selection policies for two different behaviours, multifunctional learning uses just one batch of multiagents to learn different behaviours together. The resulting multifunctional learning has not only saved the materials for behaviour conduction, but also significantly accelerated learning.



(a)



(b)

Figure 5.6: Comparison of multifunctional learning and independent learning: learning exploration and goal reaching in the simple environment E1 (averaged over fifty runs).

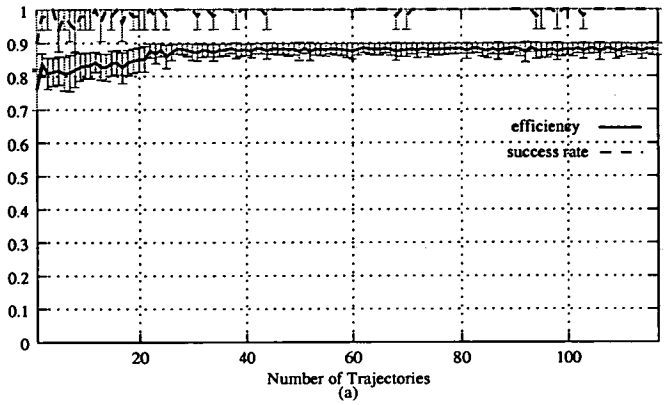
5.4.3 Learning Two Space Occupying Behaviours Together in the More Complex Environment E2

In this experiment, a fresh virtual creature which has a randomly initialised CBG and MENL is asked to learn two kinds of space occupying behaviours (exploration and goal reaching) together, but in a more complicated environment E2. Similarly to the last experiment, the virtual creature starts from fresh exploration learning in E2 and then learns a new behaviour of goal reaching by using the same behaviour model and artificial neural networks. The creature should repeat multifunctional learning of exploration and goal reaching one after the other until it is competent in both behaviours. The results of multifunctional learning will be compared with those of independent learning of exploration and goal reaching.

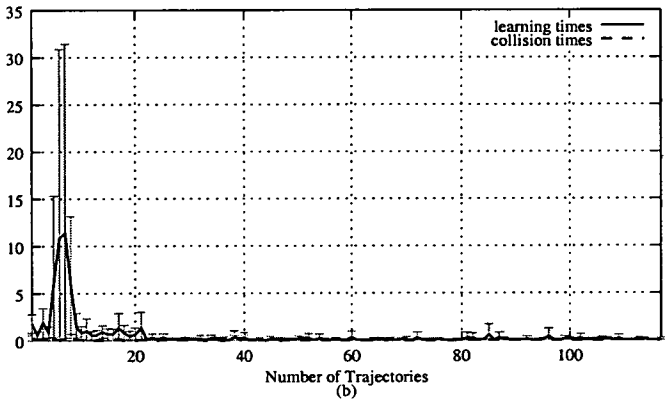
5.4.3.1 Subsequent Learning of Goal Reaching After Fresh Exploration

Section 4.4.2.2 has shown the independent exploration learning of a fresh virtual creature in environment E2. In this section, such a creature that has learned exploration in E2 is used to learn an additional behaviour of goal reaching. The creature should learn to find a way to various goal destinations from arbitrary starting points in E2, until it successfully reaches one hundred successive destinations. Every goal reaching trajectory should not go beyond 650 steps. In this experiment, each creature of the fifty previous runs of fresh exploration in E2 is used for subsequent goal reaching learning. The average results over fifty runs are shown in Figure 5.7.

The virtual creature takes a maximum of 117 trajectories to complete its subsequent goal reaching learning in fifty runs. Most of the learning is aggregated in the first 20 trajectories, in which the average learning times range from 1 to 11 in each trajectory. During the remaining 97 trajectories, the virtual creature endeavours to correct minor movement errors. In addition to a little learning on overcoming collisions (an average of 3.2 learning procedures in total), most learning is on avoiding useless turning or fixation at a place so that the creature can move towards goal destinations more directly and quickly. Even so, there are only on average 61.7 evolutionary learning procedures present in total.



(a)



(b)

Total learning times	Total collision times	Final efficiency	Final success rate
61.7 (0-125.3)	3.2 (0.8-5.6)	88.8% (87.9%-89.7%)	100% (100%-100%)

(c)

Figure 5.7: Subsequent goal reaching learning after fresh exploration in E2 (averaged over fifty runs). Each creature after fifty runs of fresh exploration in E2 is applied to reach various randomly selected goal destinations in no more than 650 steps until the creature has reached one hundred successive goal destinations successfully. The goal reaching efficiency and success rate of each trajectory averaged over fifty runs are recorded in Figure (a). The average learning times and collision times spent in each trajectory over fifty runs are shown in Figure (b). The last 100 successive and successful trajectories are not shown in Figure (a) and Figure (b). Figure (c) lists the total numbers of learning procedures and collisions required for subsequent goal reaching learning, and the goal reaching efficiency and success rate obtained at the final of the learning, all averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

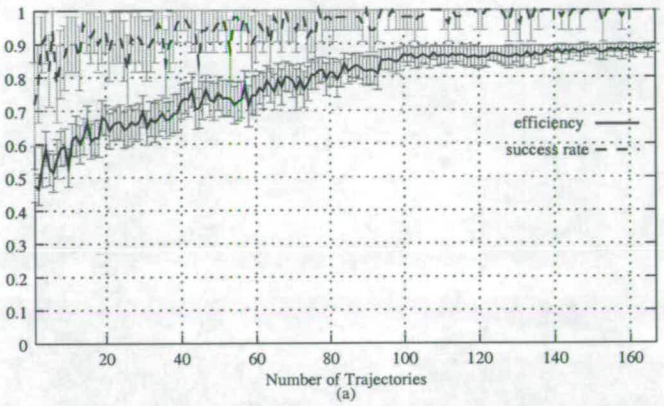
Due to the intensive learning in the early stage of the experiment, both goal reaching efficiency and success rate have exhibited an obvious increasing tendency. In the first 20 trajectories, the goal reaching efficiency is improved from 76% to 85% and the success rate is improved from 90% to 98%. In the remaining trajectories, the goal reaching efficiency is maintained around 88.6% and the success rates are kept at 100%. Through continuous adaptive learning, the virtual creature that has already learned exploration ability adjusts its functionality to goal reaching successfully.

Independent Goal Reaching Learning

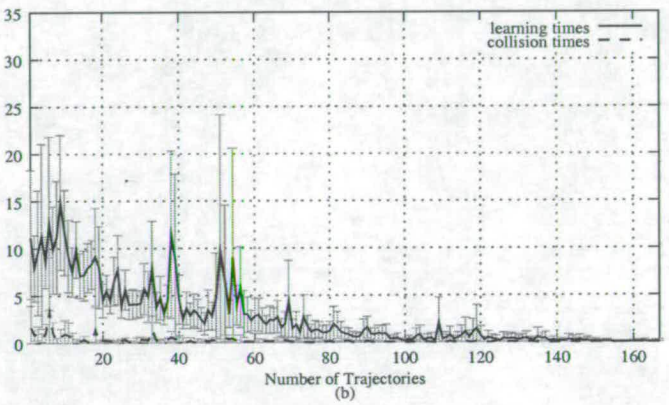
For an estimation of the performance of goal reaching learning following fresh exploration learning, an independent goal reaching experiment in environment E2 is conducted. A fresh creature equipped with a random CBG and MENL conducted goal reaching continuously until it reached one hundred successive goal destinations successfully. The average learning results over fifty runs are shown in Figure 5.8.

The fresh goal reaching learning in environment E2 is conducted quite well, however, its learning results are not so good as those of the subsequent learning. While the subsequent goal reaching learning after fresh exploration soon reaches a relatively stable learning stage with only minor errors, the independent learning suffers from intensive collisions and evolutionary adjustment for quite a long time. As a result, the goal reaching efficiency and success rate of fresh goal reaching learning oscillate back and forth continually. The total collisions and evolutionary learning procedures involved in the independent learning are almost 10 and 7 times more than those of subsequent learning. In addition, the initial goal reaching ability of independent learning is very low (e.g., 47% efficiency and 72% success rate). This contrasts to the very high starting point of subsequent goal reaching learning, whose efficiency and success rates are as high as 76% and 90% respectively.

The much better learning performance of the subsequent goal reaching learning in a more complicated environment E2 again suggests that the common space



(a)



(b)

Total learning times	Total collision times	Final efficiency	Final success rate
458.2 (372.2-544.2)	33.1 (21.1-45.1)	88.6% (87.2%-90.0%)	100% (100%-100%)

(c)

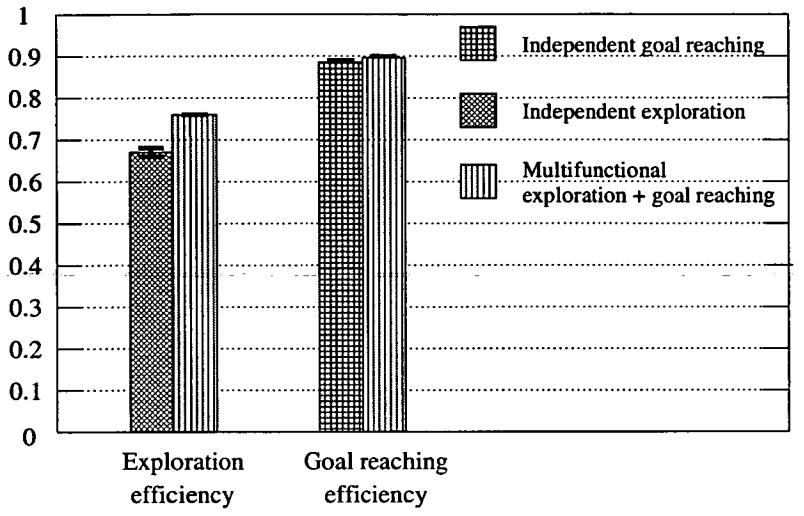
Figure 5.8: Independent goal reaching learning in E2 (averaged over fifty runs). In each run, a fresh virtual creature equipped with a randomly initialised CBG and MENL is asked to reach various random goal destinations from arbitrary places and orientations in no more than 650 steps. The learning is finished when one hundred successive goal destinations are reached successfully. The goal reaching efficiency and success rate of each trajectory averaged over fifty runs are recorded in Figure (a). The average learning times and collision times spent in each trajectory over fifty runs are shown in Figure (b). The last 100 successive and successful trajectories are not shown in Figure (a) and Figure (b). Figure (c) lists the total numbers of learning procedures and collisions required for independent goal reaching learning, and the final goal reaching efficiency and success rate obtained by the learning, all averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

occupying knowledge contained in the previously learned fresh exploration has endowed the virtual creature with a fairly good goal reaching ability and helped the following goal reaching learning to be much quicker and more efficient.

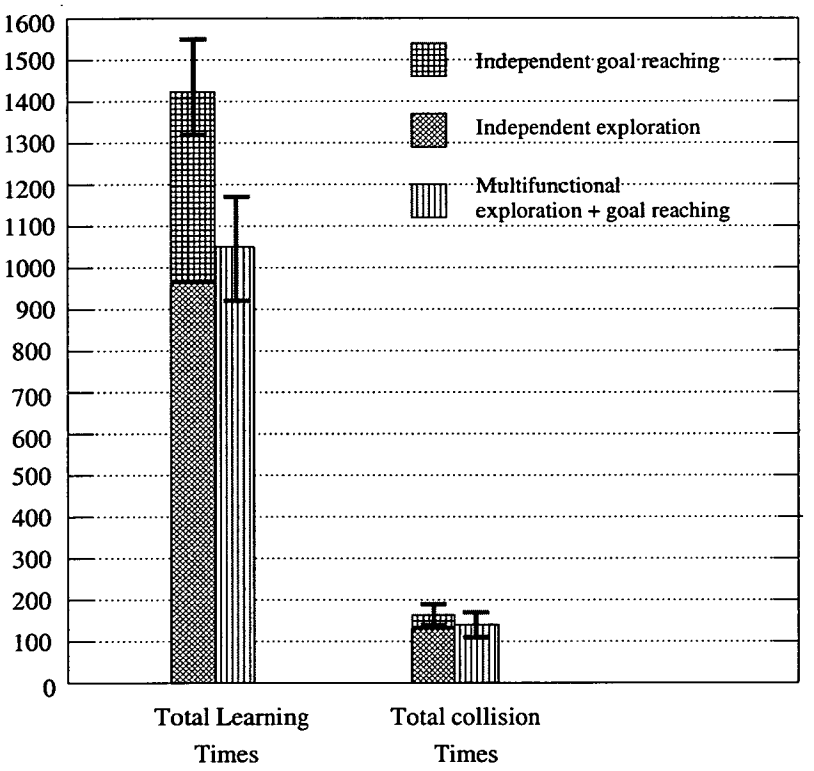
5.4.3.2 Comparison of Multifunctional Learning with Independent Learning

After learning a new behaviour in the same behaviour model CBG and MENL, the previously learned exploration is inevitably affected to some extent. This is the same as what happened in the multifunctional learning in E1, as we introduced in the last experiment. In order to learn both exploration and goal reaching well, the virtual creature has had to repeat learning these two behaviours one after another four times. The final efficiency obtained for exploration is 76.1% and the final efficiency obtained for goal reaching is 89.6%. The total learning times and collision times required are 1056 and 139.7, averaged over fifty runs.

Together with the multifunctional learning results, Figure 5.9 shows the outcome of the independent exploration and goal reaching learning in environment E2. Compared with these two independent learning procedures, the overall performance of learning exploration and goal reaching multifunctionally again exhibits its superiority. The average exploration and goal reaching efficiencies achieved by multifunctional learning are about 9 points and 1 point higher than those achieved by the corresponding single behaviour learning. Even during the learning, the collisions made in the multifunctional learning are 25 fewer than the sum of the collisions made in single behaviour learning. Most conspicuously, the total learning procedures in multifunctional learning have been much improved. While independent learning uses a sum of 1422.5 evolutionary learning procedures to learn exploration and goal reaching individually, multifunctional learning only uses 1056 learning procedures to learn both behaviours together. There are 366 learning procedures saved in total in multifunction learning. The much better overall performance of learning exploration and goal reaching together in environment E2 has shown the feasibility and efficiency of multifunctional learning, just as it did in the simple environment E1.



(a)



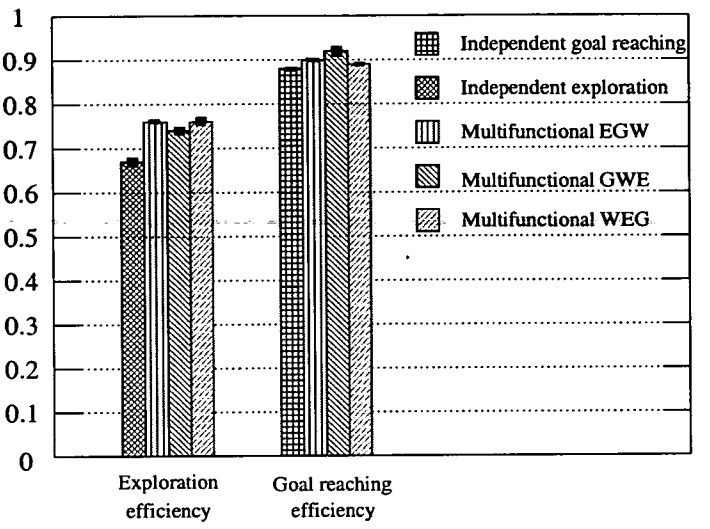
(b)

Figure 5.9: Comparison of multifunctional learning and independent learning: learning exploration and goal reaching in the complex environment E2 (averaged over fifty runs).

5.4.4 Learning Three Space Occupying Behaviours Together

In the last two experiments, the virtual creature has demonstrated its ability to learn two different but related space occupying behaviours in the same behaviour model. In this section, the virtual creature is challenged by learning three space occupying behaviours together (wandering, exploration and goal reaching) in the complicated environment E2. The creature should learn these behaviours one after another and repeat this learning periodically until the learning of every behaviour is executed free of error.

Three kinds of multifunctional learning procedures composed of different behaviour learning orders are selected, for an investigation of learning order's effect on multifunctional learning. These multifunctional learning procedures start with learning of different behaviours, which are repeated learning of Exploration+Goal reaching+Wandering (noted as EGW multifunctional learning), Goal reaching+ Wandering+Exploration (GWE), and Wandering+Exploration+Goal reaching (WEG). Every multifunctional learning procedure starts from a fresh creature with initialised CBG and MENL, and is conducted for fifty runs. The experimental results have confirmed that the virtual creature learns all three space occupying behaviours successfully in every multifunctional learning procedure. In particular, EGW repeats its learning of behaviours for at most four times in fifty runs to learn all three behaviours together. GWE repeats at most twice and WEG repeats at most five times to finish their learning in fifty runs. The total learning and collision times taken for each multifunctional learning and the resulting exploration and goal reaching efficiencies are compared with the results of three corresponding independent behaviour learning procedures, i.e., the sum of the learning and collision times required for learning three behaviours individually and the efficiencies achieved by independent exploration and goal reaching learning. Figure 5.10 shows the comparison results, averaged over fifty runs.



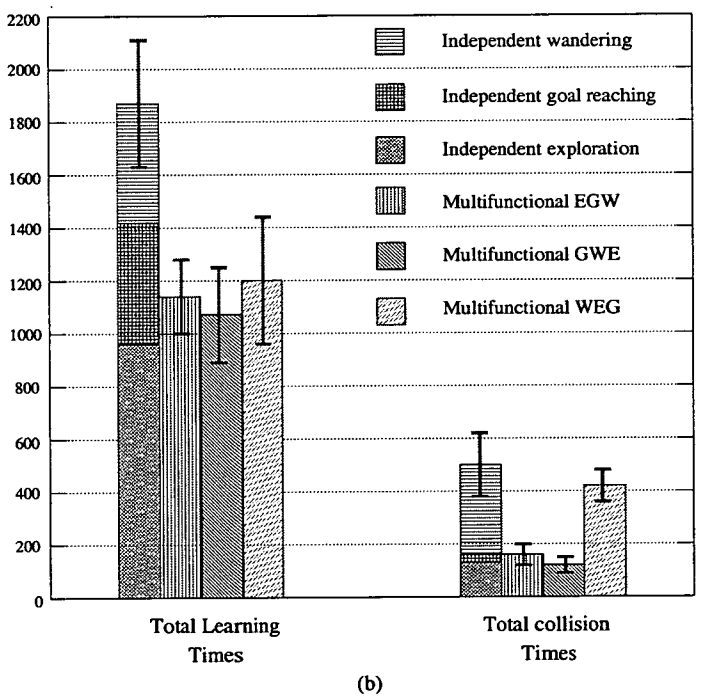


Figure 5.10: Comparison of multifunctional learning and independent learning: learning exploration, goal reaching, and wandering in environment E2 (averaged over fifty runs).

5.4.4.1 The Overall Performance of Multifunctional Learning is Better Than that of Independent Learning

Similarly to the multifunctional learning of two space occupying behaviours, all three multifunctional learning procedures on three space occupying behaviours present better overall performances than the summation of independent learning. The average learning procedures required by every multifunctional learning over fifty runs is clearly much less than the average total learning procedures required by three independent behaviour learning procedures (see Figure 5.10 (b)). Although the independent learning of wandering and goal reaching is relatively easier (446.2 and 458.2 learning steps required respectively), the learning of exploration is quite hard: nearly 964.3 steps of evolutionary learning are required. Therefore, a total of 1868.7 learning steps are presented when these three space occupying behaviours are learned independently. On the other hand, the multifunctional learning procedures of EGW, GWE and WEG take only 1143.8, 1076.4 and 1200.9 times of evolutionary learning respectively, when they have learned all three space occupying behaviours in an integrated way. Corresponding to the fewer learning procedures, the collisions made in multifunctional learning are also much fewer than those in all independent behaviour learning. Of the three kinds of multifunctional learning procedures, EGW and GWE in particular are good: their collision times are less than one third of the sum of the total collisions made in single behaviour learning. The relatively greater number of collisions made in WEG may be because of the different learning order of behaviours, as explained below. As every independent behaviour learning spends its own energy on learning collision avoidance and specific space occupying skills, multifunctional learning procedures easily learn all space occupying behaviours due to their mutual benefit.

Apart from much lower learning cost, multifunctional learning achieves similar or slightly higher exploration and goal reaching efficiencies. In both independent and multifunctional learning, WEG learning exhibits the best exploration learning ability (75.8% of the exploration efficiency), and GWE has the best goal reaching ability (92% of the goal reaching efficiency). In contrast, independent exploration

learning has the worst efficiency of 67.1% and single goal reaching learning has the lowest efficiency of 88.6%.

5.4.4.2 The Savings in Learning of Multifunctional Learning May Be Due to Mutual Optimisation Among Functional Learning Procedures

The above comparison results have shown that, when three kinds of space occupying behaviours are learned together, they have much less learning cost than learning each behaviour independently. This seems to conflict with our customary views of knowledge learning and integration, which suggest that more cost would be required on knowledge integration for different tasks. Through our experimental results, we have found that it is true that extra effort is required to integrate varied knowledge about varied behaviours (e.g., repeated learning of multiple space occupying behaviours), however, there is even greater mutual optimisation across the learning of related behaviours. It may be the mutual optimisation that saves learning time in multifunctional learning.

Figure 5.11 lists the gain/loss rates of learning a behaviour in multifunctional learning relative to independent learning of this behaviour. The positive values in the figure are gain rates, which indicate how much learning is saved when a behaviour is learned completely within a multifunctional learning procedure, compared with its corresponding independent learning; the negative values are loss rates, which show how much more learning is required to learn a behaviour within a multifunctional learning procedure. In EGW multifunctional learning, exploration learning has lost (spent) 10.2% more learning than independent exploration learning. However, the learning of goal reaching and wandering is optimised so much that about 84.4% and 97.8% learning is gained (saved), compared with their independent learning. In GWE learning, goal reaching took 13.9% more learning time, but the learning of exploration and wandering took 47.1% and 90.1% less respectively. In WEG, learning of wandering is 1.3% more than independent wandering learning. But both exploration and goal reaching learning have gained 30.2% and 81.6% learning time.

From Figure 5.11, we can see that the loss values always take place in the

Gain/loss rates	Exploration	Goal reaching	Wandering
EGW	-10.2%	+84.4%	+97.8%
GWE	+47.1%	-13.9%	+90.1%
WEG	+30.2%	+81.6%	-1.3%

Figure 5.11: Gain/loss rates of learning a behaviour in multifunctional learning relative to independent learning. The positive values in the figure are gain rates, which indicates how much learning is saved when a behaviour is learned completely in a multifunctional learning procedure, compared with its corresponding independent learning. The negative values are loss rates, which present how much more learning is required in learning a behaviour in a multifunctional learning procedure.

first learned behaviour in multifunctional learning. This may be because the first behaviour usually requires more learning in removing negative effects caused by learning of other behaviours. However, the subsequent learning of other behaviours is always much easier and faster: it always obtains a positive gain value to a certain degree. The gains in learning of the later learned behaviours may benefit from the earlier learned behaviours that have learned the common knowledge across them. Due to those gains, the total learning cost in multifunctional learning is much smaller than in independent learning procedures.

Another phenomenon we find in the gain/loss rate table is that, for the three different space occupying behaviours, their gain/loss rates within the three multifunctional learning procedures are quite different. The reason may be their different learning orders in the multifunctional learning procedures. If wandering is learned later in a multifunctional learning procedure, for instance, it can always achieve high gains from the earlier learned exploration or goal reaching (e.g., +97.8% in EGW and +90.1% in GWE). However, later learning of exploration benefits much less from the learning of the other two behaviours (e.g., +30.2% in WEG and +47.1% in GEW). These results indicate that although these three

space occupying behaviours are similar and related, their mutual effects are different. This may also explain the greater number of learning procedures and collisions made in WEG learning relative to the other two multifunctional learning procedures (see Figure 5.10). The less purposeful behaviour of wandering seems to contribute little to the subsequent learning of exploration and goal reaching which have specific goals to achieve. Mutual effects between different behaviours is one of the subjects of our ongoing research.

5.4.5 Interesting Trajectories

The above multifunctional learning experiments have exhibited many interesting trajectories which may prove the creature's robust and continuously improved space occupying ability. In this section, we briefly introduce some typical examples of these trajectories. Because these trajectories are in particular plentiful in goal reaching experiments, and because goal reaching is a broadly studied topic in the robot and virtual creature research area, interesting trajectories shown in goal reaching experiments are presented. Specifically, some trajectories display the virtual creature's ability to overcome classical canyon problems (e.g., box canyon and quasibox canyon) through its own endeavour. These trajectories have all happened when the creature has learned successful goal reaching ability. Some other interesting trajectories indicate the creature's improved goal reaching ability after further learning of exploration. Somewhat anthropomorphic language is again used in the following description, to explain the impression a human user may get from the virtual creature's behaviour.

Box Canyon

Figure 5.12 shows a goal reaching trajectory when the creature meets a trap (marked by the big circle), known as a "box canyon". The situation is similar to that described in general exploration learning in Section 4.4.2.2 and shown in Figure 4.7. But this time, the problem the creature encounters is even more difficult: in addition to finding a way out of the trap the creature should consider how to reach the goal destination in the top left corner of the environment at the

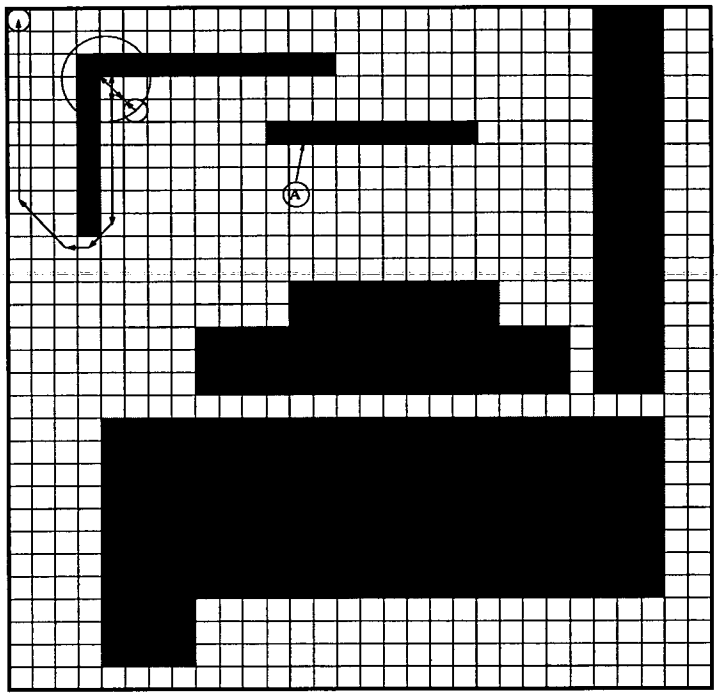


Figure 5.12: A box canyon problem. The creature, which has already learned the goal reaching ability, encounters a box canyon marked by the big circle. It turns around several times at the canyon and then moves down. The creature successfully avoids the canyon without collision and finally reaches the goal destination at the top left corner.

same time. The latter is actually conflicting with its escape behaviour. According to the experiment, the creature adjusts its direction at the corner several times, and finally goes straight down. As a result, the creature steps out of the trap, and moves towards the goal destination directly. There is neither collision nor evolutionary learning involved in this trajectory.

To make things more complicated, we extend the obstacle A (see Figure 5.12) towards the left until it is connected to the Γ-shaped obstacle (see Figure 5.13). Now, the creature is not able to move down to escape the canyon. There is only one way out of the canyon, which is to move to the right, but the creature is still commanded to reach the top left corner of the environment. In this situation, the creature adjusts its direction first and then decides to move straight down,

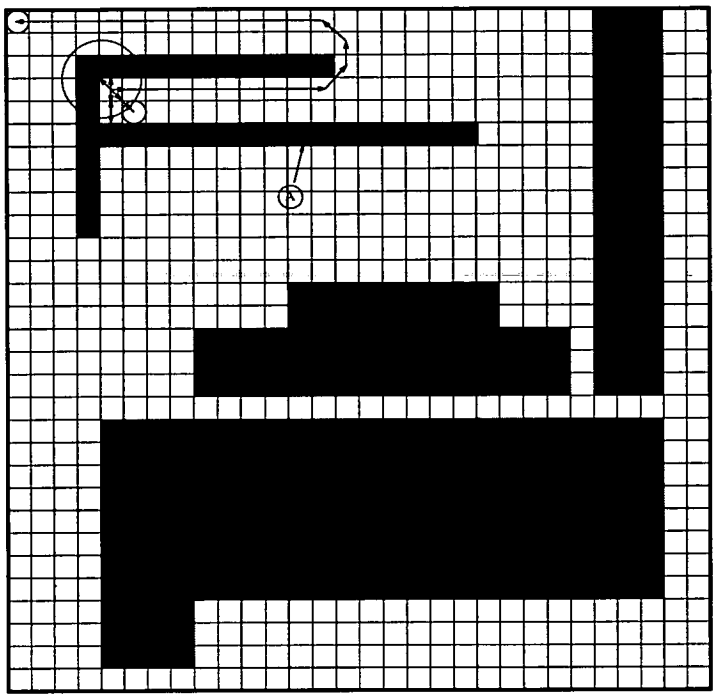


Figure 5.13: A more complicated box canyon problem. The creature, which has already learned the goal reaching ability, encounters a more complicated box canyon marked by the big circle. The creature firstly moves down and faces a big obstacle. It then moves up into the canyon again. Next, the creature turns to its right and steps out of the canyon. The creature finally reaches its goal destination at the top left corner without any problem.

as it did last time. Unfortunately, there is a big obstacle ahead. It is interesting that faced with this difficulty, the creature turns round, moves forward, and gets back to the previous corner again. In the old trap, the creature "subtly" turns to the right first and then "forges ahead without hesitation". Once the creature bypasses the obstacle, it adjusts its direction to the goal destination and moves towards it "unswervingly". In this trajectory, there is no collision or evolutionary learning present either.

Quasibox Canyon

In the trajectory presented here, the creature encounters a quasibox canyon (e.g., a box canyon with an exit). This quasibox canyon is marked by the big circle in Figure 5.14. This time, the creature starts from a random place S and tries to reach the goal location at the top right corner of the environment. During this trajectory, the creature moves towards the goal destination directly when it starts. However, it encounters a big obstacle hindering its route to the goal. The creature changes its direction to its right and moves along the obstacle in an attempt to avoid this obstacle. After a while, it steps into the quasibox canyon. Without much consideration, the creature quickly "figures out" that there is an exit at the left of the canyon and this way can get it nearer to the goal location. The creature follows this way, steps out of the canyon, and hence avoids the big obstacle successfully. Almost at the same time, the creature "finds" that its goal is just overhead and moves towards it without any hesitation.

Improved Goal Reaching Ability After Further Learning of Exploration

In multifunctional learning experiments, we found many goal reaching trajectories improved after further learning of exploration. Sometimes when the creature has finished the learning of goal reaching in multifunctional learning, it can reach goals eventually in some complicated situations but the resulting navigation trajectories are poor and unnecessarily tortuous. The solid line in Figure 5.15 shows a typical example of such tortuous trajectories. This trajectory happens when the creature has learned goal reaching for the first time after fresh exploration in E2 (see Section 5.4.3). In this example, the creature starts at the top right corner of environment E2 and eventually arrives at a destination G1. Because there is no learning during this travel, the weak goal reaching ability could not be improved this time and has to be retained in the following further exploration.

After the immediate learning of further exploration, a notable improvement on the previous goal reaching trajectory has appeared. When it starts from the top right corner of E2 and moves to G1 again, the virtual creature does not repeat the last poor and long-winded trajectory but goes towards the destination

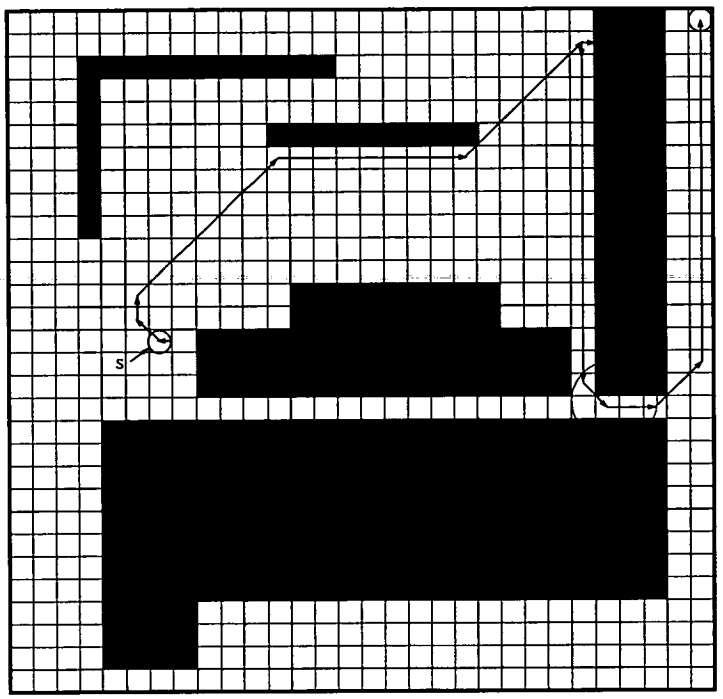


Figure 5.14: A quasibox canyon problem. When the creature learned goal reaching ability moves into the quasibox canyon marked by the big circle, it immediately turns to its right and walks along the exit to step out of the canyon. It then moves in a direct manner towards the goal at the top right corner.

directly (shown by the dashed line in Figure 5.15). The creature in fact selects the shortest route for this goal reaching navigation. Such trajectory improvement has been found in many runs of the multifunctional learning of exploration and goal reaching in environment E2. Because this goal reaching trajectory is made immediately after further exploration, we have reason to believe that the improved goal reaching ability arose from the knowledge learned from exploration. Due to the common knowledge shared between exploration and goal reaching, the creature is able to improve one space occupying ability by learning another.

The above interesting trajectories and their similarities have also been noticed in the other exploration and wandering experiments, and in environments other than E2. The creature has shown its robust problem solving ability and

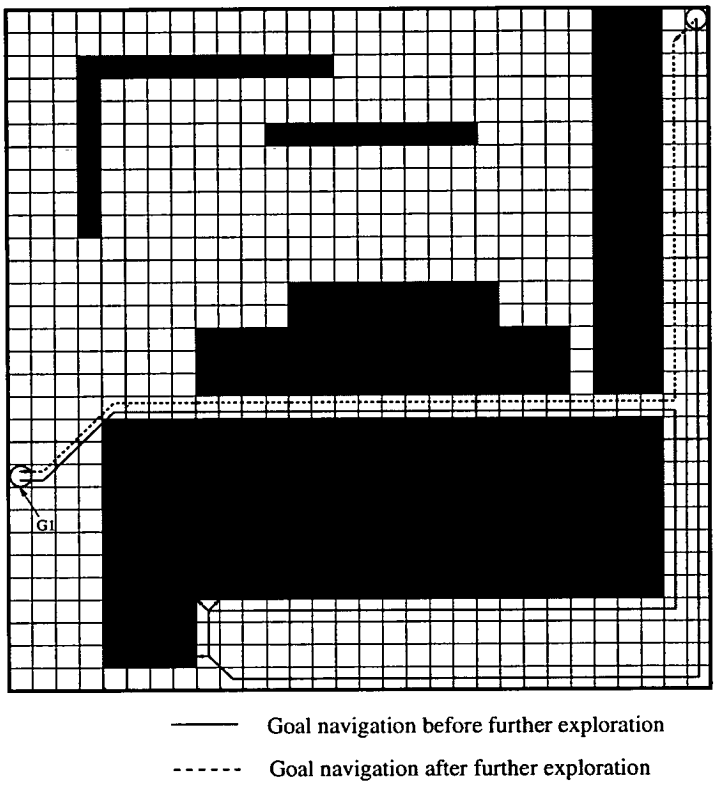


Figure 5.15: The improved goal reaching trajectory after further learning of exploration. In the multifunctional learning of exploration and goal reaching in environment E2 (see Section 5.4.3), the creature has exhibited improved goal navigation ability after further exploration learning. The solid line shows a goal reaching trajectory when the creature has finished the first learning of goal reaching after fresh exploration in E2. This trajectory starts from the right top corner and ends at a place of $G1$. Although the goal $G1$ is reached, this trajectory is unnecessarily tortuous. After it conducts the immediate second exploration learning in E2, the creature reaches the goal $G1$ directly. Such improved goal trajectories have been found in many runs of the multifunctional learning experiment in E2.

continuously improved space occupying ability throughout the experiments.

5.4.6 Summary

In this section, we have conducted a body of experiments to demonstrate that the MENL learning algorithm can achieve multifunctional learning in artificial neural networks, and when multifunctional MENL is used in the Strategy module to choose suitable motor actions for varied behaviours, the CBG model becomes multifunctional as well.

Specifically, we have utilised the CBG model combined with the MENL learning algorithm to control a virtual creature to learn multiple space occupying behaviours jointly in some unknown environments. Different behavioural motivations initiated in the Motivation module of the CBG model are embodied in different fitness functions guiding MENL to learn different action selection policies. The CBG combined with MENL has been used to learn exploration and goal reaching together in a simple unknown environment E1 and a more complex environment E2. In both environments, the subsequent learning of goal reaching based on the first learned exploration is obviously easier than learning goal reaching independently. Due to the extra learning of a new behaviour in the same behaviour model and the same artificial neural networks, the previously learned exploration is inevitably affected to some extent. However, the exploration ability is soon recovered after a few more evolutionary learning procedures. After several repetitive learning episodes of exploration and goal reaching, the CBG with MENL eventually conducts both space occupying behaviours quite well. The total cost of learning these two behaviours together is much less than the sum of learning each behaviour independently. The learned exploration and goal reaching efficiencies are similar to or even slightly better than those achieved by independent learning.

The CBG model combined with MENL has also learned three space occupying behaviours (exploration, goal reaching, and wandering) together in environment E2. Three multifunctional learning procedures with different function learning orders are conducted in this experiment. The experimental results have shown

that the overall performance of all multifunctional learning procedures have again surpassed independent behaviour learning. Because the later learned behaviours can usually benefit from the common knowledge learned by earlier behaviours, the total learning cost has been greatly reduced in multifunctional learning. Nevertheless, the reduced cost varies when multiple behaviours are learned in different orders. This suggests that multiple behaviours may have different mutual effects on each other.

During the experiments, we have found many interesting trajectories executed by the virtual creature, including the overcoming of several classical canyon problems in goal reaching and improved goal reaching trajectories after further exploration. These trajectories have shown the creature's robust and continuously improved behaviours.

5.5 Conclusion

This chapter has shown that it is possible to endow the Computational Behaviour Generation (CBG) model with multifunctional learning ability. Inspired by biological multifunctional neural networks in the brain, a multifunctional MENL is constructed, which can be dictated to by varied behavioural motivation signals to select correct motor actions for varied behaviours. These behaviours share all of the materials in the CBG model, including sensors and motors, and three modules (Movement, Programming, and Strategy) in the Computational Motor Control (CMC) system. The MENL learning algorithm is used to learn multiple action select policies in the same artificial neural networks in the Strategy module, by taking advantage of the shared common knowledge across functionally related behaviours.

The multifunctional learning ability of the CBG and MENL has been tested by several experiments. A virtual creature equipped with a CBG and MENL has been trained to learn several space occupying behaviours together. This includes learning of exploration and goal reaching together in unknown environments E1 and E2, and learning of exploration, goal reaching, and wandering together in

environment E2. Satisfactory experimental results support the claim that MENL can adaptively learn different action selection policies for multiple behaviours in the same artificial neural networks and in the same CBG model. Moreover, the overall performance of multifunctional learning is better than that of the sum of learning every behaviour independently. Compared with independent behaviour learning, multifunctional learning that learns new behaviours based on pre-learned related behaviours has not only saved material resources (artificial neural networks), but also accelerated learning speed and achieved sometimes slightly better performance when implementing behaviours.

Chapter 6

Conclusion

High fidelity virtual environments can be inhabited by virtual lives. A virtual life is a computational entity that has a lifelike visual shape and appearance and believable behavioural patterns. In order to "inhabit" its environments successfully, a competent virtual life should possess at least several important properties in behaviours. That is, the life should be autonomous to decide what to do in what circumstances based entirely on its own decisions, adaptive to adjust to changing conditions and environments and improve its behaviours accordingly, and interactive to obtain the information and resources necessary for its subsistence from the environments, other virtual lives, and even the human user.

Previous work of behavioural animation has made a breakthrough on giving virtual lives autonomy and adaptation properties to a certain degree, and releasing the lives from the very rigid, man-made behaviours. By designing several behaviour generation rules, behavioural animation has produced lifelike behaviours, which are almost impossible to create in the traditional animation approach. However, although great efforts have been made in behavioural animation, the virtual lives created so far have two serious limitations in their behaviours. First, these virtual lives cannot learn efficiently to improve their behaviours adaptively according to continuously changing situations and environments. Second, the lives do not know how to produce new behaviours by taking advantage of existing resources and pre-learned knowledge for related behaviours. These limitations inevitably restrict the autonomy and adaptation ability of virtual lives.

This thesis is concerned with the generation of believable behaviour for virtual lives. It has addressed the above two limitations in behaviour control of virtual lives by constructing an adaptive and multifunctional Computational Behaviour Generation (CBG) model and an efficient learning algorithm, a Multi-agent based Evolutionary artificial Neural network with Lifetime learning (MENL). Both CBG and MENL have taken their inspiration from the natural behaviour control mechanisms in the brain. Moreover, the design of the MENL learning algorithm has drawn on the experience of some other technologies, including evolutionary robotics, reinforcement learning, multi-agent systems, and knowledge transfer between artificial neural networks. As the CBG model provides the fundamental resources and information for adaptive and multifunctional learning, MENL obtains an increasingly improved learning ability with less human supervision and fewer constraints, and implements multifunctional learning of related behaviours in the same artificial neural networks with economy and efficiency in both space and time.

The CBG model is designed as a general and complete system. It consists of sensors and motors to collect information from and act on the virtual environments respectively. It has a Motivation module to generate behavioural motivations. It also contains a Computational Motor Control (CMC) system hierarchically composed of Strategy, Program and Movement modules to perform the whole procedure of selection, programming, and execution of motor actions for achieving behavioural motivations. The CBG model possesses the potential adaptation and multifunctionality due to its bi-directional information flows (top-down control information and bottom-up sensory feedback) and hierarchical architecture.

MENL is an evolutionary artificial neural network that can learn varied motor action selection policies for varied behaviours in the Strategy module of the CBG model. MENL utilises a batch of agents, each of which is an evolutionary artificial neural network, to cooperate and compete with each other for deciding actions to be executed. Based on the instant input of the perceived environmental information, MENL agents co-decide motor actions in various situations. These decisions

are guided by top-down commands of behavioural motivations sent from the Motivation module of the CBG system. The top-down commands are embedded in the fitness functions of agents. In addition to behavioural motivations, the general sensory feedback on executed motor actions obtained from the environment and body states is another important factor contributing to the fitness functions. As a consequence MENL agents can learn to choose suitable motor actions according to both successful and unsuccessful experience. The learning of MENL agents is continued through all of their lifetime, so the agents have continually improved decision-making ability which is adaptive and robust to varied situations and environments. By adjusting the behavioural motivations in the fitness functions, MENL agents can also be multifunctional to learn to produce multiple behaviours together. These behaviours have different objectives but share the same artificial neural networks in MENL for motor action selection and most other parts of the CBG model for action execution. Due to the general fitness function design, MENL can learn the common knowledge shared across these behaviours and transfer the common knowledge from learning of one behaviour to another. By executing motor actions selected by MENL, the CBG can generate sequences of movement to carry out single and multiple behaviours.

Successful experiments on a virtual creature have verified the adaptation and multifunctionality by learning of the CBG model combined with the MENL learning algorithm. Specifically, this virtual creature is equipped with the CBG model and MENL and required to learn several space occupying behaviours that are common and fundamental to many natural animals. The virtual creature is first asked to learn exploration in a series of unknown environments with increasing complexity. Starting from scratch, the creature should learn to explore these environments as far as possible in a limited number of steps and with attention to obstacle avoidance. Experimental results have shown that this creature has learned successful exploration in these environments without collision. Its exploration ability is similar to and sometimes slightly better than that of the hand-crafted exploration based on the availability of the places (squares) adjacent to the virtual creature. During the experiment, the creature's exploration abil-

ity is shown to be increased via MENL learning and this ability is accumulated from environment to environment. The whole successful exploration experiment is achieved by the competition and emergent cooperation among multiagents and their continuous lifetime learning.

The virtual creature commanded by the CBG model and the MENL is asked to learn multiple space occupying behaviours jointly. In particular, this creature is required to learn exploration and goal reaching together, and exploration, goal reaching and wandering together in some unknown environments. By goal reaching, we mean that the creature should reach arbitrary goal destinations in a limited number of steps without any collision. By wandering, we mean that the creature moves randomly in the environment as long as it does not hit any obstacle. In all multifunctional learning experiments, the virtual creature has successfully learned to perform multiple behaviours in the same behaviour model, and it can switch from one behaviour to another smoothly. Since new behaviours are obtained from previously learned related behaviours, the learning of subsequent behaviours is easy, economic, and computationally efficient. In addition, the overall performance of learning multiple behaviours together is better than that of the sum of learning each behaviour independently.

By means of the CBG model combined with the MENL learning algorithm, a virtual creature can possess the fundamental properties of virtual lives (i.e., autonomy, adaptation and interaction) to some extent. The virtual creature can interact with its outside (e.g., environment) to obtain necessary information and to execute its behaviours via CBG sensors and motors. Based on the interaction, the virtual creature can execute various behaviours according to its own decisions. Moreover, the creature can learn to improve its behaviours adaptively in various situations and environments, and learn to generate new related behaviours by taking advantage of the knowledge learned from previous behaviours. In consequence the virtual creature has enhanced autonomy and adaptation in behaviours. The human design on designing every detail of every behaviour can therefore be greatly reduced as well.

In the rest of the chapter, we will draw conclusions on the contributions of the

research work described in this thesis, and then discuss directions for extending this research.

6.1 Contributions

The work described here has made a number of contributions to the study of virtual lives, especially to the implementation of adaptation and multifunctionality in their behaviours by learning. These contributions are mainly embodied in the CBG model, the MENL learning algorithm, and the adaptive and multifunctional learning of the CBG model combined with MENL. These contributions are summarised below.

The Computational Behaviour Generation Model

Inspired by the natural behaviour control in the brain, the CBG model is designed with procedures of not only abstract selection, but also concrete programming and execution of motor actions. These procedures are hierarchically organised in the CBG model, so the higher levels of the CBG model can work without much consideration of elaborate implementation details of lower levels.

Not surprisingly, the CBG model is somewhat similar and related to some other behaviour models in virtual life and robot studies. However, the CBG model distinguishes itself from others by providing the fundamental support in its architecture for adaptive and multifunctional learning. The adaptation and multifunctionality of the CBG model have been proved by its successful learning of various space occupying behaviours, as shown in the above experiments.

Multi-agent based Evolutionary Artificial Neural Network with Lifetime Learning

The particular evolutionary multiagents and lifetime learning proposed in MENL are new. Multiagents hold the whole population information of every evolutionary generation, and hence increase the probability and reliability of solving problems. While learning continuously during their lifetime, multiagents have

increased adaptation to more and more situations and problems. The learning of exploration in experiments confirms that multiagents outperform the single agent with the best fitness, and multiagents with lifetime learning surpass non-improving multiagents.

The fitness function designed in MENL is not just for solving one specific problem. It contains also the information about general sensory feedback which is common to problems of the same kind. This kind of design helps choose general solutions suitable for a wide range of situations, and preserves useful common knowledge across functionally related behaviours, which is useful for learning multiple behaviours together in the same MENL. The CBG combined with MENL therefore is able to learn single and multiple space occupying behaviours in various unknown environments, as shown in the experiments.

Adaptive Learning of the CBG Combined with MENL

The adaptive learning of the CBG with MENL is achieved on-line with little human supervision and few constraints. As the CBG collects the feedback on executed actions from the environment and body states, MENL discovers useful information in this feedback and improves its decision-making ability accordingly. These processes are implemented autonomously. The resulting behaviours can therefore be adaptively improved to overcome new complicated situations without seeking help from the human designer. In addition, the learned behaviours are accumulative during its lifetime and applicable to a series of virtual environments. When the CBG model combined with MENL is used to learn space occupying behaviours, we see that the learned behaviours are adaptive to various unknown environments and the behaviours are transferred from environment to environment. Some emergent behaviours that are robust, efficient and lifelike are also found in the experiments.

Multifunctional Learning of the CBG Combined with MENL

To the author's knowledge, the CBG model combined with MENL is the first to implement multifunctional learning in the same behaviour control model, and

especially in the same artificial neural networks. With a general fitness function, MENL easily learns how to make multifunctional action selection decisions for varied behaviours, by extending the fitness function to include varied behavioural objectives. The multifunctional learning of the CBG with MENL works quite well in empirical tests, often resulting in conspicuously accelerated learning as well as improved learning results.

6.2 Future Work

Our work on the CBG model and the MENL learning algorithm is just a start. There are a number of improvements and extensions that could be done to the work described in this thesis. Some of the major issues we would like to pursue in the future are:

Improvement of the CBG Model

Although the CBG model proposed in this thesis involves a complete procedure of behaviour generation, this model only simply simulates a very small part of the natural behaviour control system. As we explained in Section 3.3, the real natural behaviour control system is a very comprehensive model involving much more complicated structures and mechanisms and numerous functions. In order to simulate complex, varied behaviours of natural creatures, the CBG model should be improved and enriched in every part and in many ways. To enable this, further study of neuroscience, ethology, psychology and some other related disciplines will be needed.

One particular improvement of our interest is to implement adaptation and multifunctionality in the CBG model at various levels. At present, we have only studied how to make the CBG model adaptive and multifunctional at the motor action selection level in the Strategy module. The concrete implementation of the Motivation, Program and Movement modules are pre-designed. In future, we will study correct and autonomous function generation in these three modules as well. Also, the function generation should be adaptive to new situations and

multifunctional to new behaviours. One possible way to do this is to use MENL, which has been proved to be adaptive and multifunctional, to learn suitable functions at different levels. In addition to study the biological mechanisms in the natural behaviour control systems further, we may need to extend and improve the MENL algorithm when necessary so that it suits new learning tasks.

Improvement of the MENL Learning Algorithm

In the learning experiments of space occupying behaviours, the space occupying abilities eventually learned are not the best. For instance, the efficiency obtained in exploration experiments is no more than 80% and the efficiency achieved in goal reaching is lower than 93%. The exploration performance is only similar to that of the hand-crafted controller which takes consideration of the information about the immediate positions of the virtual creature. However, since MENL has also shown slightly better exploration performance than the hand-crafted controller in fresh learning in environment E1 and in multifunctional learning in E1 and E2, we are confident that an improvement in the MENL learning algorithm could be made to surpass the hand-crafted design. This may be achieved by giving MENL a more efficient artificial neural network structure, a good memory mechanism, a better balance between generalisation and optimisation, and a more sensible coalition of multiagents.

Another possible improvement of the MENL learning algorithm is to extend MENL's capability in changing environments. In the experiments presented in this thesis, an environment is static once it is set up and a virtual creature is the only active entity in the environment. In future work, an environment may be dynamically changing and the human user and more than one virtual creature may move in the environment at the same time. We believe that MENL is able to deal with changing environments, because its output decisions are based on the local environmental information perceived at the current moment. Further work on this study will be conducted.

Further Theoretical and Empirical Study of Multifunctional Learning

We have empirically demonstrated that the CBG with MENL possesses multifunctional learning ability and this learning is better than independent behaviour learning. Nevertheless, a more sophisticated study of multifunctional learning is still required. In particular, although we have mentioned some possible common knowledge shared between space occupying behaviours in Chapter 3, what knowledge is exactly transferred is uncertain. We hope to study this issue in future and propose some appropriate knowledge acquisition and utilisation techniques for a better sharing of common knowledge between similar behaviours. We also expect to study the actual functions of multiagents and the mutual effects of different behaviour learning procedures in MENL's multifunctional learning. Thus work may be conducted theoretically by using automata theory to analyse the states and their activity changes of the CBG model, or empirically by executing more experiments to test agent functions and multifunctional learning functionality.

Further Study of Optimisation and Generalisation in Behaviours

In both chapters 4 and 5, we mentioned the difference and relationship between optimisation and generalisation. Generally speaking, in behaviour learning, a behaviour is expected from being general so as to suit a wide range of situations, rather than being optimal only in a limited number of situations. However, the generality of a behaviour should not prevent this behaviour being optimal when required on occasions. When a behaviour is trained to work best in some cases, its applicability to other cases should still be maintained and should not be affected. We hope to study the important issues of optimisation and generalisation in behaviour learning in future. We wish to start the study with the MENL learning algorithm that possesses a generation of homogeneous agents and a general fitness function design with particular behavioural objectives. By designing the fitness function sensibly and training MENL agents appropriately, a fitting balance between optimisation and generalisation may be achieved in behaviour generation.

Construction of a Three-dimensional Virtual Creature

The virtual creature used in this thesis is a very simple two-dimensional abstract. We wish to construct a three-dimensional graphical virtual creature in the near future. The visually lifelike body shape and appearance for the virtual creature can be designed by using popular graphical or non-graphical programming languages, such as OpenGL, C++, Visual Basic, etc. We can even create a new visual model for this virtual creature when necessary. The believable behaviours of the creature would be generated by the CBG model we proposed. The maintenance of a normal life for a virtual creature in 3D environments is generally more complicated than in 2D environments, because the creature may encounter many practical problems it cannot perceive in 2D environments, such as stereo visual information processing and more degrees of freedom in the body joints. The construction of a 3D virtual creature inhabiting 3D environments will make our research on virtual creatures more practical and nearer to real lives.

Appendix A

Statistical Techniques

The statistical techniques used in this thesis to analyse experimental data are normal statistical methods [40]. Suppose that there are N sample data $\{ x_1, x_2, ..., x_N \}$, then their mean μ is:

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}, \qquad x \in [1, N] \tag{A.1}$$

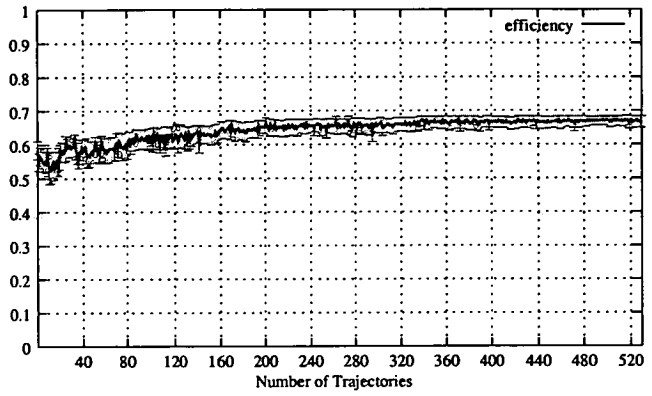
and the standard deviation σ of the sample mean (or "standard error") is:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N(N-1)}} \tag{A.2}$$

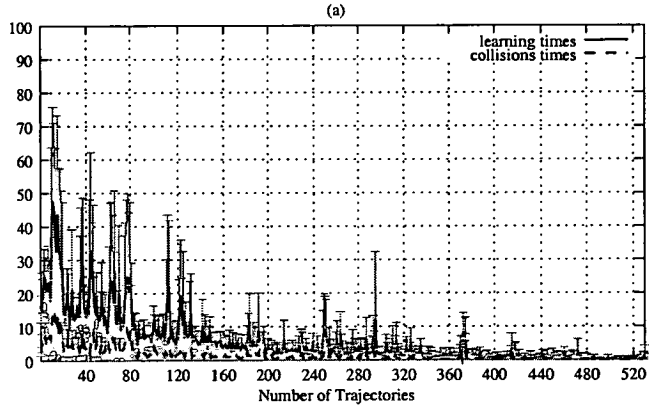
Thus, the 95 percent confidence interval about the mean is $\mu \pm 1.96\sigma$.

Appendix B

Experimental Results of Successive Exploration Learning in E3 After Fresh Exploration in E2



(a)



(b)

Total learning times	Total collision times	Final efficiency
2961.0 (2257.3-3664.7)	765.7 (575.8-955.6)	67.6% (67.4%-67.8%)

(c)

Figure B.1: Successive exploration learning in E3 after fresh learning in E2 (averaged over fifty runs). The creature learned from fresh exploration in E2 is asked to learn exploration in E3. Each creature resulting from the previous fifty runs of fresh exploration in E2 continuously explores E3 from random positions until it explores the environment successfully for one hundred successive times. Each exploration trajectory in E3 consists of 780 steps. Figure (a) shows the exploration efficiency achieved in each trajectory, and Figure (b) shows the learning times and collision times made in each trajectory, all averaged over fifty runs. The last 100 successive and successful trajectories are not shown in Figure (a) and (b). Figure (c) lists the total learning times, the total collision times, and the exploration efficiency finally achieved in this learning, averaged over fifty runs. The numbers in brackets in Figure (c) are the corresponding result ranges over fifty runs.

Bibliography

[1] Albus, J.S. (1991). Outline for a theory of intelligence. *IEEE Transactions on System, Man and Cybernetics*, 21(3):473-509.

[2] Arbib, M.A. and Liaw, J.S. (1995). Sensorimotor transformations in the worlds of frogs and robots. *Artificial Intelligence*, 72:53-79.

[3] http://www.anark.com/index.html

[4] Anderson, T.L. and Donath, M. (1990). Animal behaviour as a paradigm for developing robot autonomy. In Maes, P., editor, *Designing Autonomous Agents*, pages 145-168. Cambridge, MA. MIT Press.

[5] Andrews, J.R. (1983). *Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator.* S.M. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology.

[6] Annett, J. (1995). Motor skills. In Machintosh, N.J. and Colman, A.M., editors, *Learning and Skills.* Longman, London and New York.

[7] Aono, M. and Kunii, T. (1984). Botanical tree image generation. *IEEE Computer Graphics and Applications*, 4(5):10-34.

[8] Asada, M. (1990). Map building for a mobile robot from sensory data. *IEEE Transactions on System, Man and Cybernetics*, 20(6):1326-1336.

[9] Astheimer, P., Dai F., Gobel, M., Kruse R., Muller, S., and Zachmann, G. (1994). Realism in virtual reality. In Thalmann N.M. and Thalmann D.,

editors, *Artificial Life and Virtual Reality*, pages 189-210. John Wiley & Sons.

[10] Back, T. and Schwefel, H.P. (1993). An overview of evolutionary algorithms for parameter optimisation. *Evolutionary Computation*, 1(1):1-23.

[11] Back, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press.

[12] Bates, J. (1994). The role of emotion in believable agents. *Communications of the ACM*, 37(7):122- 125.

[13] Bates, J., Loyall, A.B. and Reilly, W.S. (1994). An architecture for action, emotion, and social behaviour. In *Proceedings of the Fourth European Workshop on Modelling Autonomous Agents in a Multi-agent World*, pages 55-68. Springer-Verlag.

[14] Becker, C., Salas, J., Tokusei, K., and Latombe, J.C. (1995). Reliable navigation using landmarks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 401-406. IEEE Computer Society Press.

[15] Belew, R.K., McInerney, J. and Schraudolph, N.N. (1991). Evolving networks: using the genetic algorithm with connectionist learning. In Langton, C.G., Taylor, C., Farmer, J.D. and Rasmussen, S., editors, *Artificial Life II*, pages 487-509. Workingham: Addison-Wesley.

[16] Beer, R.D. and Gallagher, J.C. (1990). Evolving dynamical neural networks for adaptive behaviour. *Adaptive Behaviour*, 1:91-122.

[17] Bekoff, A., Nussbaum, M.P., Sabichi, A.L. and Clifford, M. (1987). Neural control of limb coordination. I. Comparison of hatching and walking motor output patterns in normal and deafferented chicks. *Journal of Neuroscience*, 7:2320-2330.

[18] Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*. Oxford Press.

[19] http://www.bwgame.com/

[20] Blumberg, B.M. and Galyean, T.A. (1995). Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of SIGGRAPH'95*, pages 47-54. ACM Inc.

[21] Blumberg, B.M., Todd, P.M. and Maes, P. (1996). No bad dogs: ethological lessons for learning in Hamsterdam. In Maes, P., Mataric, M., Meyer, J., Pollack, J. and Wilson, S., editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on the Simulation of Adaptive Behaviour*, pages 295-304. Cambridge, MA. MIT Press.

[22] Bond, A.H. and Gasser, L. (1988). An analysis of problems and research in DAI. In Bond, A.H. and Gasser, L., editors, *Readings in Distributed Artificial Intelligence*, pages 3-35. Morgan Kaufmann.

[23] Booker, L. (1988). Classifier systems that learn internal world models. *Machine Learning Journal*, 3(3):161-192.

[24] Bonarini, A. (1997). Anytime learning and adaptation of structured fuzzy behaviours. *Adaptive Behaviour*, 5(3-4):281-315.

[25] Braitenberg, V. (1984). *Vehicles, Experiments in Synthetic Psychology.* Cambridge, MA. MIT Press.

[26] Bratman, M., Israel, D. and Pollack, M. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(4):349-355.

[27] Brooks, R.A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, VRA-2(1):14-23.

[28] Brooks, R.A. (1991). Intelligence without representation. *Artificial Intelligence*, 47:139-159.

[29] Brooks, V.B. (1986). *The Neural Basis of Motor Control.* Oxford University Press.

[30] Bryson, S. (1995). Approaches to the successful design and implementation of virtual reality applications. In Earnshaw, R.A., Vince, J.A., and Jones, H., editors, *Virtual Reality Applications*, pages 3-15. Academic Press.

[31] Burdea, G. and Coiffet, P. (1993). *Virtual Reality Technology.* Wiley-Interscience Publication.

[32] Cangelosi, A., Parisi, D. and Nolfi, S. (1994). Cell division and migration in a genotype for neural networks. *Network*, 5:497-515.

[33] Caruana, R. (1997). Multitask learning. *Machine Learning*, 28:41-75.

[34] Chao, G., Panangadan, A. and Dyer, M.G. (2000). Learning to integrate reactive and planning behaviors for construction. In Meyer, J.A., Berthoz, A., Floreano, D., Roitblat, H.L. and Wilson. S.W., Editors, *From Animals to Animats 6: Proceedings of the Sixth International Conference on the Simulation of Adaptive Behaviour.* Cambridge, MA. MIT Press.

[35] Chavas, J., Corne, C., Horvai, P., Kodjabachian, J. and Meyer, J.A. (1998). Incremental evolution of neural controllers for robust obstacle-avoidance in Khepera. In *Proceedings of the First European Workshop on Evolutionary Robotics*, pages 224-244. Springer-Verlag.

[36] Cliff, D. (1991). Computation neuroethology: a provisional manifesto. In Meyer, J.A. and Wilson, S.W., editors, *From Animals to Animats 1: Proceedings of the First International Conference on the Simulation of Adaptive Behaviour*, pages 29-40. Cambridge, MA. MIT Press.

[37] Cliff, D. (1991). The computational hoverfly: a study in computational neuroethology. In Meyer, J.A. and Wilson, S.W., editors, *From Animals to Animats 1: Proceedings of the First International Conference on the Simulation of Adaptive Behaviour*, pages 87-96. Cambridge, MA. MIT Press.

[38] Cliff, D., Harvey, I. and Husbands, P. (1993). Explorations in evolutionary robotics. *Adaptive Behaviour*, 2(1):73-110.

[39] Cohen, P.R., Atkin, M. and Hansen, E.A. (1994). The interval reduction strategy for monitoring cupcake problems. In Cliff, D., Husbands, P., Meyer, J.A. and Wilson, S., editors, *From Animals to Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour*, pages 82-90. Cambridge, MA. MIT Press.

[40] Cohen, P.R. (1994). *Empirical Methods for Artificial Intelligence.* Cambridge, MA. MIT Press.

[41] http://www.cyberlife.co.uk.

[42] Dean, J., Kindermann, T., Schmitz, J., Schumm, M. and Cruse, H. (1999). Control of walking in the stick insect: from behaviour and physiology to modeling. *Autonomous Robots*, 7:271-288.

[43] Dickinson, P.S. (1995). Interactions among neural networks for behaviour. *Current Opinion in Neurobiology*, 5:792-798.

[44] Doncieux, S. (2001). Evolution d'Architectures de Contrôle pour Animats Volants. Mémoire de stage de DEA IARFA. Université Pierre et Marie Curie.

[45] Dorigo, M. and Schnepf, U. (1993). Genetic-based machine learning and behaviour based robotics: a new synthesis. *IEEE Transactions on System, Man and Cybernetics*, 23:141-154.

[46] Eggenberger, P. (1996). Cell interactions as a control tool of developmental processes for evolutionary robotics. In Maes P., Mataric M., Meyer J., Pollack J. and Wilson S., editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on the Simulation of Adaptive Behaviour*, pages 440-448. Cambridge, MA. MIT Press.

[47] Filliat, D., Kodjabachian, J. and Meyer, J.A. (1999). Evolution of neural controllers for locomotion and obstacle-avoidance in a 6-legged robot. *Connection Science*, 11:223-240.

[48] Filliat, D., Kodjabachian, J. and Meyer, J.A. (1999). Incremental evolution of neural controllers for navigation in a 6-legged robot. In *Proceedings of the Fourth International Symposium on Artificial Life and Robotics*, pages 753-760. Oita University Press.

[49] Floreano, D. and Mondada, F. (1994). Automatic creation of an autonomous agent: genetic evolution of a neural-network driven robot. In Cliff, D., Husbands, P., Meyer, J.A. and Wilson, S., editors, *From Animals to Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour*, pages 421-430. Cambridge, MA. MIT Press.

[50] Floreano, D. and Mondada, F.(1996). Evolution of homing navigation in a real mobile robot. *IEEE Transactions on System, Man and Cybernetics*, 26(3):396-407.

[51] Floreano, D. (1993). Emergence of nest-based foraging strategies in ecosystems of neural networks. In Meyer, J.A., Roitbalt, H.L. and Wilson, S.W., editors, *From Animals to Animats 2: Proceedings of the Second International Conference on the Simulation of Adaptive Behaviour*, pages 410-416. Cambridge, MA. MIT Press.

[52] Foner, L.N. and Maes, P. (1994). Paying attention to what's important: using focus of attention to improve unsupervised learning. In Cliff, D., Husbands, P., Meyer, J.A. and Wilson, S., editors, *From Animals to Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour*, pages 256-265. Cambridge, MA. MIT Press.

[53] http://www.fujitsu.co.jp.

[54] Gelfand, I.M., Orlovsky, G.N., and Shik, M.L. (1988). Locomotion and scratching in tetrapods. In Cohen, A.H., Rossignol, S. and Grillner, S., editors, *Neural Control of Rhythmic Movements in Vertebrates*, pages 167-169. New York. Wiley.

[55] Getting, P.A. and Dekin, M.S. (1985). Mechanisms of pattern generation underlying swimming in Tritonia. IV. Gating of central pattern generator. *Journal of Neurophysiology*, 53:466-480.

[56] Getting, P.A. (1989). Emerging principles governing the operation of neural networks. *Annual Review of Neuroscience*, 12:185-204.

[57] Gomi, T. and Griffith, A. (1996). Evolutionary robotics - an overview. In *Proceedings of the IEEE Third International Conference on Evolutionary Computation*, pages 40-49. IEEE Computer Society Press.

[58] Grand, S., Cliff D. and Malhotra, A. (1997). Creatures: Artificial life autonomous software agents for home entertainment. In *Proceedings of the First International Conference on Autonomous Agents*, pages 22-29. ACM Press.

[59] Grefenstette, J.J., Ramsey, C.L. and Schultz, A.C. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning*, 5:355-381.

[60] Grefenstette, J.J. and Connie, L.R. (1992). An approach to anytime learning. *Proceedings of Ninth International Machine Learning Workshop*, pages 189-195.

[61] Grier, J.W. and Burk, T. (1992). *Biology of Animal Behaviour*. Mosby-Year Books.

[62] Gruau, F. (1994). Neural network synthesis using cellular encoding and the genetic algorithm. PhD Thesis, Ecole Normale Superieure de Lyon.

[63] Gruau, F. and Quatramaran, K. (1997). Cellular encoding for interactive evolutionary robotics. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 368-377. Cambridge, MA. MIT Press.

[64] Grzeszczuk, R. and Terzopoulos, D. (1995). Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of SIGGRAPH'95*, pages 63-70. ACM Inc.

[65] Guillot, A. and Meyer, J.A. (2000). From SAB94 to SAB2000: what's new, animat? In Meyer, J.A., Berthoz, A., Floreano, D., Roitblat, H.L. and Wilson. S.W., Editors, *From Animals to Animats 6: Proceedings of the Sixth International Conference on the Simulation of Adaptive Behaviour*, pages 364-374. Cambridge, MA. MIT Press.

[66] Gynther, I.C. and Pearson, K.G. (1989). An evaluation of the role of identified interneurons in triggering kicks and jumps in the locust. *Journal of Neurophysiology*, 61:45-57.

[67] Hallam, B.E., Halperin, J.R.P. and Hallam, J.C.T. (1995). An ethological model for implementation on mobile robots. *Adaptive Behaviour*, 3:51-80.

[68] Harvey, I., Husbands, P. and Cliff, D. (1993). Issues in evolutionary robotics. In Meyer, J.A., Roitbalt, H.L. and Wilson, S.W., editors, *From Animals to Animats 2: Proceedings of the Second International Conference on the Simulation of Adaptive Behaviour*, pages 364-374. Cambridge, MA. MIT Press.

[69] Harvey, I., Husbands, P. and Cliff, C., Thompson, A. and Jakobi, N. (1997). Evolutionary robotics: the Sussex approach. *Robotics and Autonomous Systems*, 20(2-4):205-224.

[70] Holland, J.H. (1986). Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In Michalski, R.S., Carbonell, J.G. and Mitchell, T.M., editors, *Machine Learning, an Artificial Intelligence Approach VII*, pages 593-624. Morgan Kaufmann.

[71] Honda, H. (1971). Description of the form of trees by the parameters of the tree-like body: effects of the branching angle and the branch length on the shape of the tree-like body. *Journal of Theoretical Biology*, 31:331-338.

[72] Houk, J.C. (1980). Homeostasis and control principles. In Mountcastle, V.B., editor, *Medical Physiology*. St. Louis. Mosby Co.

[73] Humphrys., M. (1996). Action selection methods using reinforcement learning. In Maes P., Mataric M., Meyer J., Pollack J. and Wilson S., editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on the Simulation of Adaptive Behaviour*, pages 135-144. Cambridge, MA. MIT Press.

[74] Ijspeert, A.J., Hallam, J., and Willshaw, D. (1998). From lampreys to salamanders: evolving neural controllers for swimming and walking. In Pfeifer, R., Blumberg, B., Meyer, J.A., and Wilson, S.W., editors, *From Animals to Animats 5: Proceedings of the Fifth International Conference on the Simulation of Adaptive Behaviour*, pages 390-399. Cambridge, MA. MIT Press.

[75] Ijspeert, A.J., Hallam, J. and Willshaw, D. (1998). Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology. Department of Artificial Intelligence Research Paper 876, University of Edinburgh.

[76] Ijspeert, A.J. and Kodjabachian, J. (1999). Evolution and development of a central pattern generator for the swimming of a lamprey. *Artificial Life*, 5(3):247-269.

[77] Kaelbling, L.P. (1990). *Learning in Embedded Systems.* PhD thesis, Stanford University.

[78] Kaelbling, L.P. and Moore, A.W. (1996). Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237-285.

[79] Kandel, E.R., Schwartz, J.H. and Jessell, T.M. (1995). *Essentials of Neural Science and Behaviour.* Prentice Hall Inter. Inc.

[80] Kikuchi, K. and Hara, F. (1998). Evolutionary design of morphology and intelligence in robotic system using genetic programming. In Pfeifer, R., Blumberg, B., Meyer, J.A., and Wilson, S.W., editors, *From Animals to Animats 5: Proceedings of the Fifth International Conference on the Simulation of Adaptive Behaviour*, pages 540-545. Cambridge, MA. MIT Press.

[81] Kimura, H., Akiyama, S. and Sakurama, K. (1999). Realization of dynamic walking and running of the quadruped using neural oscillator. *Autonomous Robots*, 7:247-258.

[82] Kodjabachian, J. and Meyer, J.A. (1998). Evolution and development of modular control architectures for 1-D locomotion in six-legged animats. *Connection Science*, 10:211-237.

[83] Kodjabachian, J. and Meyer, J.A. (1998). Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects. *IEEE Transactions on Neural Networks*, 9(5):796-812.

[84] Koza, J.R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA. MIT Press.

[85] Krogh, B.H. and Thorpe, C.E. (1986). Integrated path planning and dynamic steering control for autonomous vehicles. In *IEEE International Conference on Robotics and Automation*, pages 1664-1669. IEEE Computer Society Press.

[86] Kurtz, C. (1991). The evolution of information gathering: operational constraints. In Meyer, J.A. and Wilson, S.W., editors, *From Animals to Animats 1: Proceedings of the First International Conference on the Simulation of Adaptive Behaviour*, pages 376-381. Cambridge, MA. MIT Press.

[87] Lazanas, A. and Latombe, J.C. (1992). Landmark-based robot navigation. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 816-822. Cambridge, MA. MIT Press.

[88] Lee, W.P., Hallam, J. and Lund, H.H. (1997). Applying genetic programming to evolve behaviour primitives and arbitrators for mobile robots. In *Proceedings of the IEEE Fourth International Conference on Evolutionary Computation*, pages 501-506. IEEE Computer Society Press.

[89] Lohmann, K.J. and Lohmann, C.M.F. (1996). Detection of magnetic field intensity by sea turtles. *Nature*, 380:59-61.

[90] Lohmann, K.J. and Lohmann, C.M.F. (1994). Detection of magnetic inclination angle by sea turtles: a possible mechanism for determining latitude. *Journal of Experimental Biology*, 194:23-32.

[91] Lund, H.H., Hallam, J. and Lee, W.P. (1997). Evolving robot morphology. In *Proceedings of the IEEE Fourth International Conference on Evolutionary Computation*, Invited paper. IEEE Computer Society Press.

[92] Maes, P. (1990). Situated agents can have goals. In Maes, P., editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 49-70. Cambridge, MA. MIT Press.

[93] Maes, P. (1995). Artificial life meets entertainment: lifelike autonomous agents. *Communications of the ACM*, 38(11):108-114.

[94] Maes, P. (1994). Modeling adaptive autonomous agents. *Artificial Life*, 1(1-2):135-162.

[95] Mahadevan, S. and Connell, J. (1992). Automatic programming of behaviour-based robots using reinforcement learning. *Artificial Intelligence*, 55(2-3):311-365.

[96] Mataric, M. (1992). Integration of representation into goal-driven behaviour-based robot. *IEEE Transactions on Robotics and Automation*, 8(3):304-312.

[97] Mataric, M.J. (1994). Reward functions for accelerated learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 181-189. Morgan Kaufmann.

[98] Mataric, M.J. and Cliff, D. (1996). Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems*, 19(1):108-114.

[99] Mataric, M.J., Zordan, V.B. and Williamson, M.M. (1999). Making complex articulated agents dance, an analysis of control methods drawn from robotics, animation, and biology. *Autonomous Agents and Multi-agent Systems*, 2(1):23-44.

[100] http://www.maxis.com.

[101] Měch, R. and Prusinkiewicz, P. (1996). Visual models of plants interacting with their environment. *Proceedings of SIGGRAPH'96*, pages 397-410. ACM Inc.

[102] Meyer, J.A. and Guillot, A. (1991). Simulation of adaptive behaviour in animats: review and prospect. In Meyer, J.A. and Wilson, S.W., editors, *From Animals to Animats 1: Proceedings of the First International Conference on the Simulation of Adaptive Behaviour*, pages 2-14. Cambridge, MA. MIT Press.

[103] Meyer, J.A. and Guillot, A. (1994). From SAB90 to SAB94: Four years of animat research. In Cliff, D., Husbands, P., Meyer, J.A. and Wilson, S., editors, *From Animals to Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour*, pages 3-11. Cambridge, MA. MIT Press.

[104] Meyer, J.A. (1998). Evolutionary approaches to neural control in mobile robots. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 35-40. IEEE Computer Society Press.

[105] Meyer, J.A., Doncieux, S., Filliat, D. and Guillot, A. (to appear) Evolutionary approaches to neural control of rolling, walking, swimming and flying animats or robots. In Duro, R.J., Santos, J. and Graa, M., editors, *Biologically Inspired Robot Behaviour Engineering*. Springer-Verlag.

[106] Michel, O. (1995). An artificial life approach for the synthesis of autonomous agents. In *Proceedings of the European Conference on Artificial Evolution*, pages 220-231. Springer-Verlag.

[107] Minsky, M. (1988). *The Society of Mind*. Simon & Schuster, New York.

[108] Mondada, F. and Floreano, D. (1995). Evolution of neural control structures: some experiments on mobile robots. *Robotics and Autonomous Systems*, 16:183-195.

[109] Moriarty, D.E. and Miikkulainen, R. (1994). Evolving neural networks to focus minimax search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1371-1377. Cambridge, MA. MIT Press.

[110] Moriarty, D.E. and Miikkulainen, R. (1996). Evolving obstacle avoidance behaviour in a robot arm. *Machine Learning*, 22:11-32.

[111] Moriarty, D.E. and Miikkulainen, R. (1998). Forming neural networks through efficient and adaptive co-evolution. *Evolutionary Computation*, 5(4):373-399.

[112] Moriarty, D.E., Schultz, A.C and Grefenstette, J.J. (1999). Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:199-229.

[113] Mortin, L.I. and Stein, P.S.G. (1989). Spinal cord segments containing key elements of the central pattern generators for three forms of scratch reflex in the turtle. *Journal of Neuroscience*, 9:2285-2296.

[114] Mller, R., Maris, M. and Lambrinos, D. (1999). A neural model of landmark navigation in insects . *Neurocomputing*, 26-27:801-808.

[115] Nolfi, S. (1997). Evolving non-trivial behaviours on real robots: a garbage collecting robot. *Robotics and Autonomous Systems*, 22(3-4):187-198.

[116] Noser, H., Renault, O., Thalmann D. and Thalmann, N.M. (1995). Navigation for digital actors based on synthetic vision, memory and learning. *Computers and Graphics*, 19(1):7-19.

[117] Oppenheimer, P. (1986). Real time design and animation of fractal plants and trees. *Proceedings of SIGGRAPH'86*, pages 55-64. ACM Inc.

[118] Orido, G., Ulivi, G. and Vendittelli, M. (1998). Real-time map building and navigation for autonomous robot in unknown environments. *IEEE Transactions on System, Man and Cybernetics*, 28(3):316-333.

[119] Panerai, F., Metta, G. and Sandini., G. (2000). Adaptive image stabilization: a need for vision-based active robotics agents. In Meyer, J.A., Berthoz, A., Floreano, D., Roitblat, H.L. and Wilson. S.W., Editors, *From Animals to Animats 6: Proceedings of the Sixth International Conference on the Simulation of Adaptive Behaviour.* Cambridge, MA. MIT Press.

[120] Paredis, J. (1991). The evolution of behaviour: some experiments. In Meyer, J.A. and Wilson, S.W., editors, *From Animals to Animats 1: Proceedings of the First International Conference on the Simulation of Adaptive Behaviour*, pages 419-426. Cambridge, MA. MIT Press.

[121] Parker, G.B. and Karen J.L. (2000). Punctuated anytime learning for evolutionary robotics. *World Automation Congress Proceedings*, pages 268-273.

[122] Pearson, K.G. (1993) Common principles of motor control in vertebrates and invertebrates. *Annual Review of Neuroscience*, 16:265-297.

[123] http://www.pfmagic.com.

[124] Pratt, L.Y., Mostow, J. and Kamm, C.A. (1991). Direct transfer of learned information among neural networks. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 584-589. AAAI Press.

[125] Pratt, L.Y. (1993). Discriminability-based transfer between neural networks. In Moody J.E., Hanson S.J. and Lippmann R.P., editors, *Advances in Neural Information Processing Systems V*, pages 204-211. Morgan Kaufmann.

[126] Prescott, T.J. (1996). Robot spatial learning: insights from animal and human behaviour. In *IEE Workshop on Self Learning Robots.*

[127] Prusinkiewicz, P., Hammel, M., Měch R. and Hanan J. (1995). The artificial life of plants. *SIGGRAPH'95 course notes on Artificial Life*, pages 1-38. ACM Inc.

[128] Prusinkiewicz, P. and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*, Springer-Verlag, New York.

[129] Rao, A. and Georgeff, M. (1992). An abstract architecture for rational agents. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 439-449. Morgan Kaufmann.

[130] Rechenberg, I. (1973). *Evolution Strategies.* Frommann-Holzboog, Stuttgart.

[131] de Reffye, P., Edelin, C., Francon, J., Jaeger, M., and Puech, C. (1988). Plant models faithful to botanical structure and development. *Proceedings of SIGGRAPH'88*, pages 151-158. ACM Inc.

[132] Renault, O., Thalmann, N.M., and Thalmann, D. (1990). A vision-based approach to behavioural animation. *The Journal of Visualisation and Computer Animation*, 1(1):18-21.

[133] Reynolds, C. (1987). Flocks, herds and schools: a distributed behavioural model. *Computer Graphics*, 21(4):25-34.

[134] Reynolds, C. (1994). Evolution of corridor following behaviour in a noisy world. In Cliff, D., Husbands, P., Meyer, J.A. and Wilson, S., editors, *From Animals to Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour*, pages 401-410. Cambridge, MA. MIT Press.

[135] Roitblat, H.L. (1991). Cognitive action theory as a control architecture. In Meyer, J.A. and Wilson, S.W., editors, *From Animals to Animats 1: Proceedings of the First International Conference on the Simulation of Adaptive Behaviour*, pages 444-450. Cambridge, MA. MIT Press.

[136] Salomon, R. (1996). Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions: a survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39(3):263-278.

[137] Salomon, R. (1997). The evolution of different neuronal control structures for autonomous agents. *Robotics and Autonomous Systems*, 22(3-4):199-213.

[138] Schultz, A.C. and Grefenstette J.J. (1992). Using a genetic algorithm to learn behaviours for autonomous vehicles. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, pages 739-749. American Institute of Aeronautics & Astronautics.

[139] Schultz, A.C. (1994). Learning robot behaviours using genetic algorithms. *Intelligent Automation and Soft Computing: Trends in Research, Development, and Applications*, pages 607-612. TSI Press, Albuquerque.

[140] Schultz, A.C. and Grefenstette, J.J. (1996). Robo-shepherd: Learning complex robotic behaviours. *Robotics and Manufacturing: Recent Trends in Research and Applications*, 6:763-768.

[141] Schwefel, H.P. (1995). *Evolution and Optimum Seeking.* Wiley, New York.

[142] Selverston, A.I., Panchin, Y.V., Arshavsky, Y.I. and Orlovsky, G.N. (1997) Shared features of invertebrate central pattern generators. In Stein P.S.G., Grillner S., Selverston A.I. and Stuart D.G., editors, *Neurons, Network and Motor Behaviour*, pages 105-107. MIT Press.

[143] Sharkey, N.E. (1991). Connectionist representation techniques. *Artificial Intelligence Review V*, 5:143-167 .

[144] Sharkey, N.E. and Sharkey, A.J.C. (1992). Adaptive Generalisation and the Transfer of Knowledge. Working report R257, Center for Connection Science, University of Exeter, UK.

[145] Sharkey, N.E. (1997). Artificial neural networks for coordination and control: the portability of experiential representations. *Robotics and Autonomous Systems*, 22(3-4):345-360.

[146] Singh, S.P. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323-339.

[147] Simpson, G.G., Pittendrigh C.S. and Tiffany L.H. (1959). *Life: An Introduction to Biology.* Routledge and Kegan Paul Ltd.

[148] Sims, K. (1994). Evolving virtual creatures. In *Proceedings of SIGGRAPH'94*, pages 15-22. ACM Inc.

[149] Sims, K. (1994). Evolving 3D morphology and behaviour by competition. In *Proceedings of Artificial life IV*, pages 28-39. Cambridge, MA: MIT Press.

[150] Sutton, R. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3:216-224.

[151] Sutton, R. (1990). Integrated architectures for learning, planning and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216-224. Morgan Kaufmann.

[152] Sutton, R. (1991). Planning by incremental dynamic programming. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 353-357. Morgan Kaufmann.

[153] Terzopoulos, D. (1998). Vision and action in artificial animals. In Harris L.R. and Jenkin M., editors, *Vision and Action*, pages 250-276. Cambridge University Press.

[154] Thalmann, D. (1998). Artificial life of virtual humans, In *SIGGRAPH'98 course 22, Session 5.* ACM Inc.

[155] Thalmann, D. (1995). Applications of virtual humans in virtual reality. In Earnshaw, R.A., Vince, J.A. and Jones, H., editors, *Virtual Reality Applications*, pages 271-282. Academic Press.

[156] Thrun, S. and Mitchell, T.M. (1996). Lifelong robot learning. *Robotics and Autonomous Systems*, 15:25-46.

[157] Thrun, S. (1996). *Explanation-Based Neural Network Learning: A Lifelong Learning Approach.* Kluwer Academic Publisher.

[158] Thrun, S. (1998). Learning maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21-71.

[159] Trullier, O., Wiener, S.I., Berthoz, A. and Meyer, J.A. (1997). Biologically based artificial navigation systems: review and prospects. *Progress in Neurobiology*, 51:483-544.

[160] Trullier, O. and Meyer, J.A. (2000). Animat navigation using a cognitive graph. *Biological Cybernetics*, 83(3):271-285.

[161] Tyrrell, T. (1993). *Computational Mechanisms for Action Selection.* PhD. Thesis, Centre for Cognitive Science, University of Edinburgh, UK.

[162] Tu, X. and Terzopoulos, D. (1994). Artificial fishes: physics, locomotion, perception, behaviour. In *Proceedings of SIGGRAPH'94*, pages 43-50. ACM Inc.

[163] Vaario, J., Onisuka, A. and Shimohara, K. (1997). Formation of neural structures. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 214-223. Cambridge, MA. MIT Press.

[164] Walker, A., Hallam, J. and Willshaw, D. (1993). Bee-havior in a mobile robot: The construction of a self-organizing cognitive map and its use in robot navigation within a complex, natural environment. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1451-1456. IEEE Computer Society Press.

[165] Walker, M.M. (1999). Magnetic position determination by homing pigeons. *Journal of Theoretical Biology*, 197:271-276.

[166] Wang, F. and Mckenzie, E. (1998). Virtual life in virtual environments. Technical report ECS-CSG-44-98, Department of Computer Science, The University of Edinburgh.

[167] Wang, F. and Mckenzie, E. (1998). General navigation in unknown environments. Technical report ECS-CSG-45-98, Department of Computer Science, The University of Edinburgh.

[168] Wang, F. and Mckenzie, E. (1999). A multi-agent based evolutionary artificial neural network for general navigation in unknown environments. In *Proceedings of the Third International Conference on Autonomous Agents*, pages 154-159. ACM Inc.

[169] Wang, F. and Mckenzie, E. (1999). Multifunctional learning of a multi-agent based evolutionary artificial neural network with lifetime learning. In *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 332-337. IEEE Computer Society Press.

[170] Watkins, C.J.C.H. (1989). *Learning from Delayed Rewards*. Ph.D. thesis, University of Cambridge, England.

[171] Watkins, C.J.C.H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279-292.

[172] Wimann, J.M., Meyrand, P., and Marder, E. (1991). Neurons that form multiple pattern generators: identification and multiple activity patterns of gastric/pyloric neurons in the crab stomatogastric system. *Journal Neurophysiology*, 65:111-122.

[173] Werner, G.M. (1994). Using second order neural connections for motivation of behavioural choices. In Cliff, D., Husbands, P., Meyer, J.A. and Wilson, S., editors, *From Animals to Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour*, pages 154-163. Cambridge, MA. MIT Press.

[174] Whitley, D., Dominic, S., Das, R. and Anderson, C.W. (1993). Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13:259-284.

[175] Wilson, S.W. (1994). ZCS: a zeroth level classifier system. *Evolutionary Computation*, 2(1):1-18.

[176] Wyatt, J., Hoar, J. and Hayes, G. (1998). Design, analysis and comparison of robot learners. *Robotics and Autonomous Systems*, 24(1-2):17-32.

[177] Yamauchi, B. and Beer, R. (1993). Sequential behaviour and learning in evolved dynamical neural networks. *Adaptive Behaviour*, 2(219-246).

[178] Yamauchi, B. and Beer, R. (1994). Integrating reactive, sequential and learning behaviour using dynamical neural networks. In Cliff, D., Husbands, P., Meyer, J.A. and Wilson, S., editors, *From Animals to Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour*, pages 382-391. Cambridge, MA. MIT Press.

[179] Yao, X. and Liu, Y. (1998). Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on System, Man and Cybernetics*, 28(3):417-425.