Evolving A Sense Of Valency

Theodoros Damoulas

Master of Science

School of Informatics

University of Edinburgh

2004

Abstract

This thesis investigates the evolution of a mechanism able to associate valencies to the effect of behaviours executed by organisms. The sense of valency so far was incorporated into artificial intelligent systems as a pre-designed reward function or as a given property of the problem under consideration. In many cases, the design of such a function is not possible due to instabilities of the environment or limitation of the available knowledge on the solution space.

Three models are presented to attempt to overcome those limitations, by either evolving the reward function or directly the valency of behaviours. The models operate within a Darwinian evolutionary framework, and two of them also encompass the combination of reinforcement learning (RL) methods, mainly the temporal-difference (TD) Q-Learning algorithm.

Our results demonstrate that all three models can account for correct valency associations in a stable environment, and that the resulting reward function strongly depends on the fitness function and on the discount factor. However, for the case of unstable environments, only the two models that incorporate a learning mechanism have managed to adapt. Those using evolution only, failed to do so. We claim that the performance of our two successful models that incorporate learning is unaffected by valency instabilities and that this is due to the separation between genotype and phenotype.

Acknowledgements

I would like to thank Gillian Hayes for supervising me this summer, for her valuable help and ideas that kept me as close to the track as possible. I would also like to thank Ignasi Cos Aguilera for his constant support, proof-reading skills and long chats in FH and Tim Taylor for his GA expertise.

Many thanks to my flatmates, (ex and current) for their understanding and great company throughout this year, to the people from down-south and up-north for their friendship (and their “cultural differences”) and also to the people from the FH B11 lab for keeping me awake during the late hours.

Last but not least, I would like to thank my family that although separated, can still give me things I crave for.

Thanks to Adonis and Despoula for the things they gave me from their hart.

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.

(*Theodoros Damoulas*)

Nikos Kazantzakis (1883 - 1957)

“I am a weak, ephemeral creature made of mud and dream. But I feel all the powers of the universe whirling within me. Before they crush me, I want to open my eyes for a moment and see them. I set my life no other purpose. I once set out from a dark point, the Womb, and now I proceed to another dark point, the Tomb. A power hurls me out of the dark pit and another power drags me irrevocably toward the dark pit. And I strive to discover how to signal my companions before I die, how to give them a hand, how to spell out for them in time one complete word at least, to tell them what I think this procession is, and toward what we go. And how necessary it is for all of us together to put our steps and hearts in harmony.”

——————— “Askitiki”, 1927.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

This thesis investigates the origin and evolution of the sense of valency on artificial agents and provides a modelling framework for testing and evaluation. The sense of valency has been defined as "the ability to identify the utility of behaviours executed"; how "good or bad" are the effects of actions.

In the majority of the problems faced by modern AI the sense of valency, when needed, is offered through the system designer [Sutton and Barto, 2002]. In the case of supervised learning, the "supervisor" is directly evaluating, hence assigning a valency on the behaviours executed by the agent. Unsupervised learning on the other hand does not require such an ability but cannot also produce biologically plausible and adaptive behaviours. Finally, reinforcement learning, which could be placed in between, makes use of a rewarding signal in order to evaluate the learning process. However, that reward function as well is a result of a system designer.

Intelligent behaviour needs a mechanism for dynamically learning valency and such a mechanism is present in the animal brain through the dopamine system which is involved in various behavioural functions [Daw, 2003]. It is speculated[1] that the neurotransmitter is responsible for identifying and describing rewarding experiences. The evolution of the signal of reward and its interpretation is the core of our study.

[1] See [http://www.columbia.edu/ jh299/] for related publications

Our aim is to overcome the need of a pre-defined sense of valency and follow the direction of evolution towards a biologically plausible explanation of this phenomenon. We explore different ways of evolving valency through various models, that account for different theoretical proposals, and evaluate them in varying-stability environments.

1.2 Motivation

The motivation for this dissertation is the exploration of reward functions for adaptive autonomous agents. We argue that the reward function cannot be pre-designed if the agents are supposed to be really autonomous. The work was also inspired from previous related work done by Batali and Grundy [Batali and Grundy, 1996].

From a biological perspective, it is still arguable whether or not such a valency mechanism has been evolved or it is based on a hard-wired element which defines "goodness" or "badness". In this study we follow the evolutionary scenario and attempt to provide an insight to that perspective as well. This two-way relationship, from biology to the agent and opposite, is our motivation framework that led to this study.

1.3 Approach

The approach adopted is to combine reinforcement learning (RL) methods and genetic algorithms (GA) in order to create models that examine the contribution of both Learning and Darwinian Evolution into the phenomenon of valency. We concentrate on three models that either evolve the reward function and use Q-Learning to develop valency associations, or they directly evolve the sense of valency. The models are then tested and evaluated on stable and unstable environments.

1.4 Structure

The rest of the dissertation is structured in 5 chapters. The following chapter provides a detail description of the valency problem together with a general overview of the methodology. The Evolutionary theories are presented and criticised, and the main

concepts of reinforcement learning are introduced. Past relevant studies are briefly presented and their contribution toward this study is emphasised.

Chapter 3 presents analytically all the model architectures of the agent and the structure of the environment. The common elements between the models are presented together and a separate account is given for every model together with standard settings and the individual algorithm.

Chapter 4 includes the results of our experiments for all cases. The results are presented with the same order to the model architecture description on the different performances of the models. Finally, Chapter 5 and 6 include our discussion on the results, our conclusions from this study and future work that has been proposed.

Chapter 2

The Valency Problem

This chapter introduces the valency problem and the related work. A general theoretical background is discussed, from the notion and importance of valency, evolution and learning to their computational counterparts. Previous work in this direction is described synoptically, emphasising their particular characteristics and major results. Finally, our topic of research and a general description to the approach are introduced.

2.1 A Sense of Valency

Intelligence has certain common characteristics across all the organisms that possess it. One of them is that the primary goal of most (if not all) behaviours is self-preservation and survival. Even complex behaviours can be explained by the need for survival and it is arguable if there is a behaviour that is not based on the goal of self-preservation.

However, self-preservation requires the agent to have a sense of a certain amount of what is good and bad. If *all* our ancestors had considered hitting their heads on the walls of the cave as a positive action towards their preservation, we would probably not exist as a species anymore. In this respect, it could be argued that the feeling of pain would have suggested to them not to do it.

We suggest that this would not have been the case and the feeling of pain would have been associated as something pleasant if the action was considered good. Feelings can be seen strictly as an affective - thus evolutionary, association. For example, the

feeling of wind blowing in your face is pleasant for you because it has been valued as something harmless for your survival (unless you have had bad experiences with strong winds). Furthermore, associations are subjected to variation over ones lifetime according to experience.

We argue that those associations between effects of actions and their perceived quality (good or bad) are the *sense of valency*. The ability to identify the quality or valency of one's actions and effects is a major need for survival and something which can be accredited to feelings of pain or pleasure. Nevertheless, those feelings and what they actually mean to each individual are just by-products of thousands of years of experience and evolution which have resulted in a physiological response to the association between the effect and what its "goodness" is.

Another important aspect of valency is that it allows for a common scale to be applied to all actions and their effects. In every moment we make choices among a variety of available actions and we select the one we expect to be the most beneficial. To be able to do so, there must be a mechanism that allows a comparison to be made between all the different effects of actions. That mechanism is valency and it is not something that has been hard-wired into our nature but instead it evolved to the present state.

The evolution of that sense of valency and how it can be modelled is the primary objective of this study. Two important shaping processes are considered; learning and evolution. Evolution will provide the basis for the sense of valency to be passed along generations and learning will be used to explain the knowledge of valency that is gained through the lifetime of an organism. Hence evolution is concerned with the large time-scale (over generations) while learning is directly applied on the small time-scale (lifetime). The next section introduces a theoretical basis for evolution, learning and their interaction.

2.2 Evolution and Learning

2.2.1 Darwinian evolution

The present theory of evolution - modern evolutionary synthesis - is largely based on Darwin's theory published as *On the Origin of Species* [Darwin, 1859]. This relatively recent theory can be traced back to Anaximander (610-ca. 546 BC) and his perception on the evolutionary descent of man from animal. However, Darwin was the first to actually explain important aspects of that process such as natural selection and fitness and also to propose a theory of the actual mechanism. The current stream of thought is yet incomplete but in general terms it has been accepted from the scientific community as a plausible scientific explanation for the complexity of life.

Natural selection is the process by which nature (the environment) promotes and "selects" the fittest organism. If an organism has, due to random mutations on its genetic material, a new ability that is useful and increases its probability of reproduction then it will be the fittest among the rest of its species and it will pass on its genes to more organisms of the next generation. Hence that new ability (encoded in its genes) will spread through the population because of the fact that the organisms that have it in their genes will be the fittest.

Fitness is defined as the relative reproduction probability of an organism within a species. This selective pressure of the environment encourages the evolution of the species from simpler units to complex ones. It is directly connected to sexual selection since the fittest organism is more likely to achieve reproduction. Darwin's theory introduced those two basic concepts that together with Mendel's work [Mendel, 1866] and the discovery of DNA during the last century, provides a better overview on evolution.

2.2.2 Lamarckian Evolution

Another theory of evolution was developed, well before Darwin's work, by Lamarck, [Lamarck, 1809]. This theory was based on the inheritance of acquired characteristics which is the opposite of the accepted Darwinian proposal. Lamarck believed that characteristics acquired by an organism during its lifetime can be passed directly to the next generation leading to inheritable change in species over time. Although his

work is considered a great contribution to the evolutionary theory, it is proven that experience and learning through an organism's lifetime is not passed along to the next generation through genetic modification.

A typical example of the Lamarckian approach is to consider the long neck of the giraffe as a characteristic that has gradually developed due to reaching out for food (experience by stretching leading to genetic modification). The Darwinian answer is that it is all due to selective pressure. The giraffes that evolved randomly a longer neck gained higher fitness (able to collect more food) and so their genes spread through the population. The main difference is that natural selection substitutes the need posed by Lamarck for parents to pass on their learned abilities (phenotype) through genetic modification (genotype).

2.2.3 The Baldwin Effect and the Hiding Effect

Both the Baldwin and the Hiding effects are related to the interaction of learning and evolution and are in a sense two opposite phenomena. That interaction is present in our study and both effects influence the evolution of valency in a way that is described next.

Baldwin Effect

Baldwin was the first to concentrate on the important aspect of learning and its effect on the evolutionary process. His major contribution to evolution theory is the description of a phenomenon, the interaction of learning and evolution, known as the *Baldwin Effect* [Baldwin, 1896]. This phenomenon is that learning can guide evolution without the Lamarckian mechanism of inheritance of acquired characteristics.

The process by which this is achieved is by simply applying the natural selection rules over the learning cycle. The organism that can learn a useful trait in its lifetime (ontogenetic variations) will be fitter than others that cannot. If there is also a high *cost of learning*, meaning that the process of learning has a high fitness cost, then organisms that can reduce that cost by modifications of their genes - modifications that make the learning easier - will be fitter. This is the procedure by which learning "decides" the

usefulness of genetic variations.

The standard interpretation of the Baldwin Effect is that it consists of two steps. The first step is the fitness increase of the population due to the effect that learning has on the phenotype. Learning improves the performance of the phenotype using lifetime experience. When a high cost of learning is present, it is an advantage for a chromosome to have encoded information in its alleles that can help learning. Hence, by the procedure of natural selection such chromosomes will have a higher fitness and thus spread through the population. The encoding of useful information (for the lifetime learning) into the genotype is the second step of the Baldwin Effect.

Hiding Effect

The opposite phenomenon is also caused by the learning cycle and is something that has a negative or "hiding" effect on the genetic evolution. When the cost of learning is low (i.e small fitness cost) and also the learning procedure is effective, then different organisms with different genes (genotype) can exert the same behaviour (phenotype) and hence genetic evolution is not encouraged. The genetic differences between organisms are "hidden" underneath the commonality of their phenotype.

In Table 2.1 the differences between the Baldwin and the Hiding effect are summarised with respect to evolution.

Type	Cost of Learning	Evolution of Genotype
Baldwin Effect	High cost	Encouraged
Hiding Effect	Low cost	Discouraged

Table 2.1: The characteristics of the Baldwin and Hiding Effect

2.3 Computational Evolution and Learning

Having described the underlying theoretical basis of evolution and its interaction with learning, we present in this section their computational counterparts. Evolution is modelled using a typical genetic algorithm that evolves populations of chromosomes and

guides their refinement by the use of the genetic operators crossover and mutation. The learning is achieved by standard machine learning algorithms that operate in the reinforcement learning framework.

2.3.1 Genetic Algorithm

Genetic Algorithms (GAs) were first implemented in the 1960s by John Holland and his colleagues at the University of Michigan. Since then they became the standard method of simulating evolution and also a very successful algorithm for solving optimisation problems. In the last years the term GAs has been used to describe various evolutionary computation methods that resemble the original algorithm.

The design of the GA we are using in this study follows closely on the definition of a simple GA given in [Mitchell, 1998]. Our algorithm consists of the basic genetic operations of crossover, mutation and inversion on a population of chromosomes that are scored each time according to a fitness function of the environment. In Figure 2.1 we present the typical GA cycle based on a simple example.

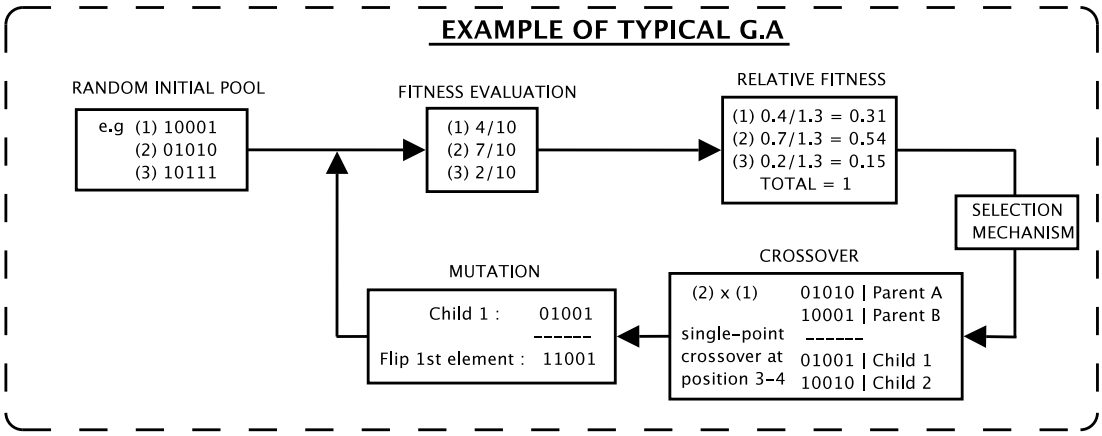


Figure 2.1: A typical GA example

The GA cycle starts from the creation of the initial population of chromosomes with randomly initialised binary alleles. The next step is the evaluation of each chromosome with respect to a fitness function and then the scaling of that evaluation on the

basis of the average fitness of the population. Based on the relative fitness a selection is made according to a roulette wheel selection method. The selected chromosomes undergo a reproduction step that combines their alleles to build the new generation of chromosomes and finally random mutation (in that case flipping the bit) is applied and the cycle repeats.

Based on that example we can briefly define the various genetic terms :

- *Chromosome* - Every chromosome is a representation or encoding of the potential solution.
- *Population* - The total number of chromosomes present in every generation.
- *Initial Pool* - It is the initial randomly created population that is used to "seed" the solution space.
- *Fitness & Relative Fitness* - The fitness of a chromosome is given by a function that encodes knowledge of the environment. The relative fitness is the fitness of the chromosome over the average fitness of the population.
- *Roulette Wheel Selection* - A selection method according to which each chromosome's probability of reproduction is analogous to its relative fitness. It is called this because it resembles a roulette wheel where each slice is the probability of reproduction for each chromosome. Bigger slices imply fitter individuals and hence large probabilities of reproduction.
- *Single-point Crossover* - The procedure by which two chromosomes can produce their offspring. One point is randomly chosen on the length of the chromosomes and at that point the chromosomes exchange parts and produce two new chromosomes.
- *Mutation* - The process in which randomly chosen alleles change their genotype. In a binary chromosome this is done by flipping the element and in a non-binary by replacing the number with a random one or by changing the existing number by a small amount.

- *Inversion* - An operation that flips the whole chromosome. The inversion has a very low probability of occurring and it is used to retain variability within the population and to account for the weakness of the single-point crossover to reorder a contiguous section of the chromosome (linkage problem, see [Mitchell, 1998]).

2.3.2 Reinforcement Learning

Learning provides the mechanism by which the sense of valency can be learnt throughout the life of an organism and affect the evolution of that sense over generations. Reinforcement Learning (RL) is a method of learning based on reinforcements that an agent (the modelling equivalent of an organism) receives from the environment. It is a biologically plausible method of learning that is based on a numerical reward signal in order to map situations to actions.

The basic elements of a RL problem are the agent and the environment. Their interaction is the core of the problem and it is also what the agent attempts to model in order to improve its perception of the environment. The learning is achieved by making use of the information for state transitions (starting state and end-state), action performed and reward collected as can be seen from Figure [1] 2.2.

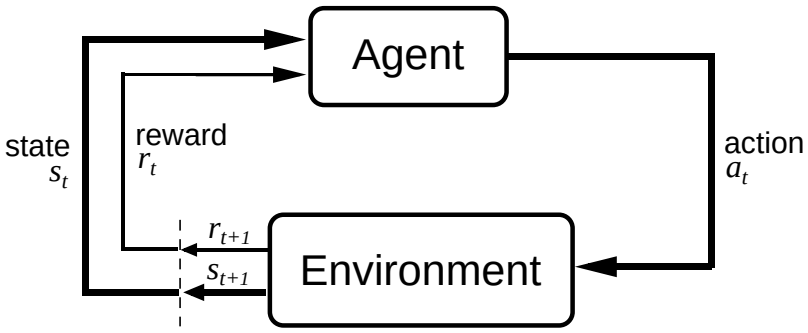


Figure 2.2: Agent-Environment Interaction [Sutton and Barto, 2002]

[1] After permission from Prof. A. G. Barto

The main concepts underlying the RL framework are :

- *State* s_t, s_{t+1} - The state representation is what the agent perceives from the environment as the effect of his actions.
- *Action* a_t - The actions that the agent can perform on the environment.
- *Policy* - Policy is the mapping from states to actions. It is the overall behaviour of the agent and it is strongly connected to the way exploration of new states or exploitation of already known states is balanced.
- *Value Function* - The value function describes the relative value of states and it is dependent not only on the specific state but also on the following states after that. It describes, in a sense, how good are states in the long run.
- *Model* - For specific RL methods a model of the environment is necessary. However this will not be the case in our study.
- *Reward Function* r_{t+1} - The most important aspect of RL is the reward function. It is the parameter that defines the goal and provides the feedback loop to the agent regarding the effect of his actions on the environment. The reward is in the core of the sense of valency since it symbolises the feeling of “pleasure” or “pain” that the agent will receive after every action.

In the majority of the problems that RL techniques were used in the actual reward function is pre-designed and incorporated into the problem. Consider for example the maze in Figure 2.3. The agent receives a negative reinforcement of -1 for every time-step it spends in the maze and a positive reward of +1 when it exits. That configuration has proven to be successful but it is due to the design of the reward function that pushes the agent towards the goal.

2.4 Related Work

In this section we present related work that has proven useful to our study. The exact problem of modelling the evolution of a sense of valency has not been examined

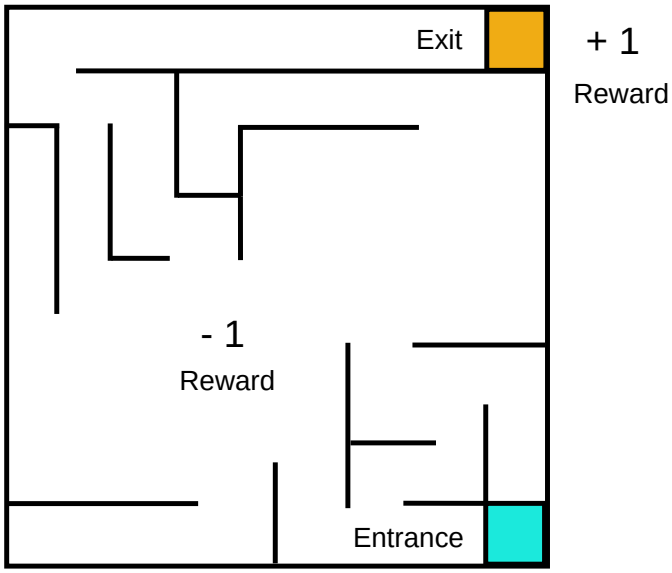


Figure 2.3: Maze example

directly and in detail in the past, but different studies have contributed towards that direction. Some of that work is synoptically presented here and it is divided into three separate sections according to the specific aspects of valency that each one is addressing. First presented is the work related closely to the essence of valency and the nature of reward, followed up by studies on the Baldwin effect and finally an inspection of the interaction between learning and evolution.

2.4.1 Studies on Valency and Reward

An important contribution in the direction of the problem is reported by [Batali and Grundy, 1996]. In their study, a motivation system is co-evolved with the learning of the behaviours it evaluates and it is the analogue to the sense of valency that informs how good or bad are certain actions in certain states. Their work, although managing to capture the essence of valency, does not report any conclusions on the actual mechanism of valency and it only concludes that it can be done.

In the study, a “valency system” is implemented in the form of a separate neural network that is used to provide a reinforcement signal on which the learning of the

behaviour-selection is based. That behaviour selection is performed via a second neural network that chooses actions given states that the agent encounters. Both neural networks are a 3-layer architecture and are formed by evolving their synaptic weights.

A different perspective on the problem of valency is offered through the work of [Shizgal, 1997]. In that study the neurological basis of utility (analogous to valency) is explored through brain-stimulation experiments in rats. Their results indicate that certain aspects of the reinforcement (electrical stimulation and sucrose), like intensity and rate, are important for the utility assigned to the reward by the rat. By varying those aspects the rats reversed their preference of one stimulant over the other as if they had selected the larger of the two rewards. Therefore they conclude "this is further evidence that natural and artificial rewards are assessed using a *common scale* of utility".

That common scale is a characteristic of valency that enables different effects of actions to be arrayed on a common dimension (pleasure-pain). The results support their hypothesis that certain myelinated axons (Medial Forebrain Bundle) encode parameters of the reinforcement in such a way that a common scale of valency/utility can be used. Their work and conclusions pin down the valency phenomenon in a neurobiological framework.

Another study related to the problem of valency has examined the operation of dopaminergic neurons and its representation through quantitative theories of adaptive optimising control [Schultz et al., 1997]. They use a standard temporal-difference (TD) rule in order to model the operation of those neurons that are responsible for encoding expectations about reward. The TD error was used as a prediction error signal for the reward and the model was used to make claims about the operation of those neurons.

One of the important conclusions is that "the reward value associated with a stimulus is not static" and that it depends on the *internal states* of the organism and the previous experience with that stimulus. They also conclude that dopamine systems may be much more precise in passing temporal information than previously thought. Although an explanation of how a such a sophisticated system of reward came to being is not provided, it can only be explained with evolutionary means.

2.4.2 Studies on the Baldwin Effect

The Baldwin effect has been simulated extensively over the last years and it has raised a lot of interest in the scientific community. The first simulation of the effect was reported by [Hinton and Nowlan, 1987]. Their results, based on a simple neural network model, identified the Baldwin effect and the smoothing of the evolutionary search-space due to the learning procedure. The smoothing of the surface is performed from learning by creating a zone of increased fitness around the highest fitness peak and the effect is limited to "needle-in-the-hay-stack"problems where it is hard to search for the solution.

Since then many researchers have dealt with the Baldwin effect in various simulated problems, [Harvey, 1996, Turney, 1996, Suzuki and Arita, 2000, Suzuki and Arita, 2003] and the effect is present in many studies relating evolution and learning. The main advance made through this work was the identification of the three steps (two steps initially) of the Baldwin effect and the necessary conditions. As it was pointed out by [Mayley, 1996], high evolutionary cost of learning is a necessary condition for the second step of the Baldwin effect to occur.

2.4.3 Studies on the Interaction Between Learning and Evolution

Studies of a different nature but related to the interaction of Learning and Evolution, contributed as well towards our models. Specifically the work done by [Mclean, 2001] was used for comparison in the unstable environments that were tested and also for constructing part of our models. The work consisted of comparing the performance of Darwinian, Lamarckian and hybrid models in stable and unstable environments using neural network architectures. The conclusions are that the hybrid model performs better overall since it combines the advantages of the other two models and also, a point to which we will come back later, that in unstable environments, although the hybrid still performs better, its performance is still dropping every time a change in the environment occurs.

Finally, other studies from [Ackley and Littman, 1991, Spier and McFarland, 1998, Miglino et al., 1995, Nolfi, 1994, Nolfi, 1999, Winter, 1998] provided useful compar-

ison with other models and reinforcement learning methods and also different views on the learning and evolutionary framework. The valency problem underlies many of those studies and proves to be a very basic element of intelligence that could explain many aspects of animal behaviour.

2.5 Approach Adopted

Taking into consideration the past work that has been done towards the problem of valency, we present the approach that is adopted towards addressing that issue. First, some important characteristics of valency and of the problem are reported and then a theoretical background for our models is presented in order to justify and clarify the approach adopted.

Problem Characteristics

According to the findings from previous work, some characteristics of valency are:

- A single dimension of valency that serves as a common scale for effects of various actions. Different actions have various effects on different aspects of an agent's architecture but valency should provide a common measure that can allow for comparison.
- The reward should not be static but depending on the internal states of the organism. On different states the same effect should be able to have different valency.
- The evolution of valency should follow the accepted Darwinian version and the learning can only influence evolution through the Baldwin effect not through Lamarckian mechanisms.
- The effect of actions should change over time, in certain occasions, in order to examine if a right sense of valency can be maintained.

Our Approach

Our approach consists of studying how an agent architecture can use evolution of rewards (internal to the agent) in order to evolve over generations correct associations between action effects and their relative "utility". In other words, the attempt is to evolve the sense of valency by using evolution of reward functions. Other agents will evolve to have wrong associations and prefer to perform actions that are "bad" for them (according to our definition of what is bad) and others will evolve a right sense of valency.

We are not simply evolving or learning a "correct" behaviour selection but we are evolving the sense of valency according to which an agent prefers certain effects of actions over others and hence learns to choose those actions. However, if it prefers the wrong effects then it will end up choosing the wrong actions and hence have a low fitness. A Darwinian-based evolution will provide the means to explore the build-up to a correct sense of valency and an unstable environment to test the behaviour of our models when drastic changes occur.

Chapter 3

Model Architecture

In this chapter, we describe the models used in our study, we analyse the different architecture characteristics and settings that were tested, and justify the necessity of those architectures in order to model *valency*. Three models are presented that account for the modeling of valency in different contexts. The first model is examining the learning and evolution framework, the second is focusing only on evolution and the last one is using again learning and evolution but through a neural network architecture for the prediction of reward.

The common characteristics of the agent and the environment between the models precede the analysis of each individual model. Finally, we describe the energy decay function and the unstable environment implementation. The base model, which directly evolves the reward values that are being used to learn the valency of actions, can be seen in Figure 3.6. The other two models present an alternative approach to the problem, first by directly evolving the valency of actions in Figure 3.7 and finally, by evolving synaptic weights of a neural network that computes the reward for the learning cycle as it can be seen in Figure 3.8.

3.1 General Considerations

As it was mentioned in the previous chapter, various models of agent-environment interaction in the context of evolution and learning have been studied in the past. Dif-

ferent actions, effects of actions, states and goal-states have been designed for stable and unstable environments. Our approach attempts to strip down the extra complexity added by those various settings by concentrating on the essence of actions, effects and states instead of having a specialised environment where the agent can perform specific actions such as “eat”, “walk”, etc. or make use of specific environment objects. The main consideration for the model was to start from a simple, well-defined architecture that would capture the essence of the problem.

The model characteristics allow such an abstract version of agent-environment interaction. The agent is composed of internal drives which are abstractions of biological needs. These guide towards certain actions and are expressed as a set of internal states. Those internal drives and their possible transitions define the available actions for the agent, and their effects with respect to the agent. For example, the need for a specific action “eat” would emerge from a drive signalling energy and nutrition levels. Transitions on that drive underline the effect of the action which is to increase the level of energy by eating, or the effect of another action that causes loss of energy.

Our models are initially deterministic in the sense that actions cause a certain effect on the drives with probability 1 (e.g. choosing action A always increases the level of drive by 0.1). The introduction, later on, of an energy decay function causes a relative stochasticity to be applied on the models, uncoupling the agent from the environment since now an action is not always guaranteed to have the desired effect.

Since evolution and learning are the main aspects of evolving a sense of what is “good” or “bad” for an agent, they are core. Evolution is achieved through a typical genetic algorithm (GA) and the learning is based on reinforcement learning techniques (RL). Generations of agents are evolved towards an optimal target which is to satisfy their drive(s) by constantly performing the action which is beneficial, according to the state they are in. The successful agents will have by evolution the right associations between an action and how good this action is for them.

The following sections introduce the different architectures. Three different models, with one or two drives, on deterministic or relative stochastic environments and on stable or unstable environments, have been proposed.

3.2 Environment and Agent

In this section we present the characteristics of the environment and the agent that are common to the different models that are described in the rest of the chapter.

3.2.1 Drives

Each agent consists of its drive(s) and its specific chromosome. The number of drives is the same for all agents in the population and each drive is divided into 10 discrete levels (cf. Figure 3.1). The current level of a drive defines quantitatively the extent to which a drive has been satisfied. In the case of energy for example, 1.0 signals energy satiation. Drive values range between 0.1 and 1.0

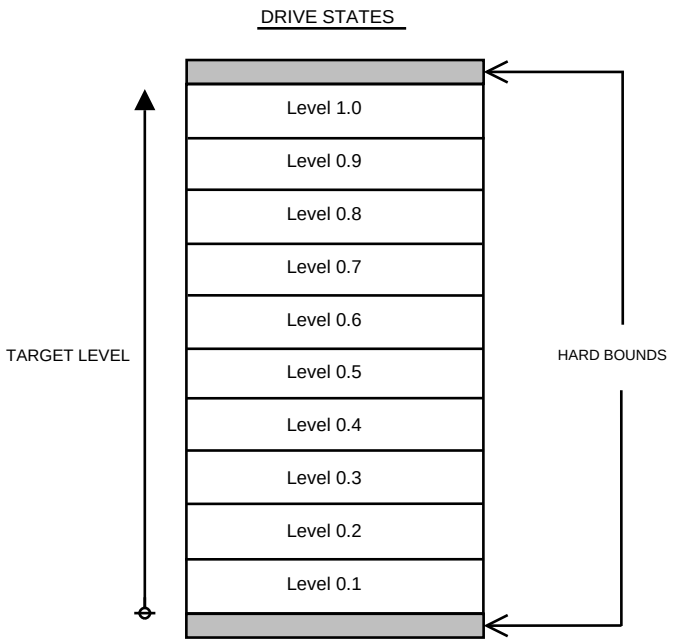


Figure 3.1: The Drive

The hard bounds of the drive are the maximum and minimum limits permitted. An action which has an effect of surpassing those limits is restricted so that the effect on the drive is to remain on the previous level. This is an ad-hoc ensuring that all agents survive the learning cycle (the other solution would be to kill any agent reaching

beyond those limits) since their genetic material might prove useful in the future. Every drive has been discretized to 10 states from 0.1 to a maximum of 1.0. The convention used was that the level of the drive corresponds to the level of satiation, hence a level of 1.0 for any of its drives is the goal that the agent must achieve.

As was mentioned before, the agent also has a chromosome. The chromosome in every model is the defining (except when learning is involved) information for the behaviour of the agent in every state and it is evolved through generations of agents in order to ensure that a correct behaviour emerges. The specific role of the chromosome is different from one model to the other and will be further described in the analogous sections.

When more than one drive is present in the agent, the dimensionality of the problem increases only linearly as opposed to exponentially. This was achieved by considering behaviour selection on every drive independent of the states that the rest of the drives were in (i.e. if behaviour A was selected for Drive 1 this would be on the basis of its own state and independent of whether Drive 2 was in state 0.1 or 1.0). Having now selected one behaviour per drive, we combine them with a winner-take-all (WTA) approach, on the basis of which behaviour has the highest valency, in order to decide on the dominant behaviour which will be executed as it can be seen from Figure 3.2. In a way, behaviours (possible actions per given state) are competing within a drive and overall, drives compete to decide which drive's selected behaviour will be executed and reinforced (if learning is present).

If learning is present, the behaviour executed will have an effect on both drives and it will also be reinforced based on both effects (if there is no learning then the effect on the rest of the drives is left unused since there is no reward to create a feedback). Hence in that case, the behaviour selection for Drive A is dependent on the **effect** the behaviour has on all drives but at the same time it only depends on the **state** of Drive A. For example consider the case where Drive 1 is on level 0.3 and Drive 2 is on level 0.6. If behaviour A is selected for Drive 1, executed (preferred over Drive 2 selected behaviour) and it has a positive effect on both Drives then it will receive good reinforcements from both Drives. Which means that next time Drive 1 is on the same state, that behaviour will be more likely to be re-selected but **independently** of the

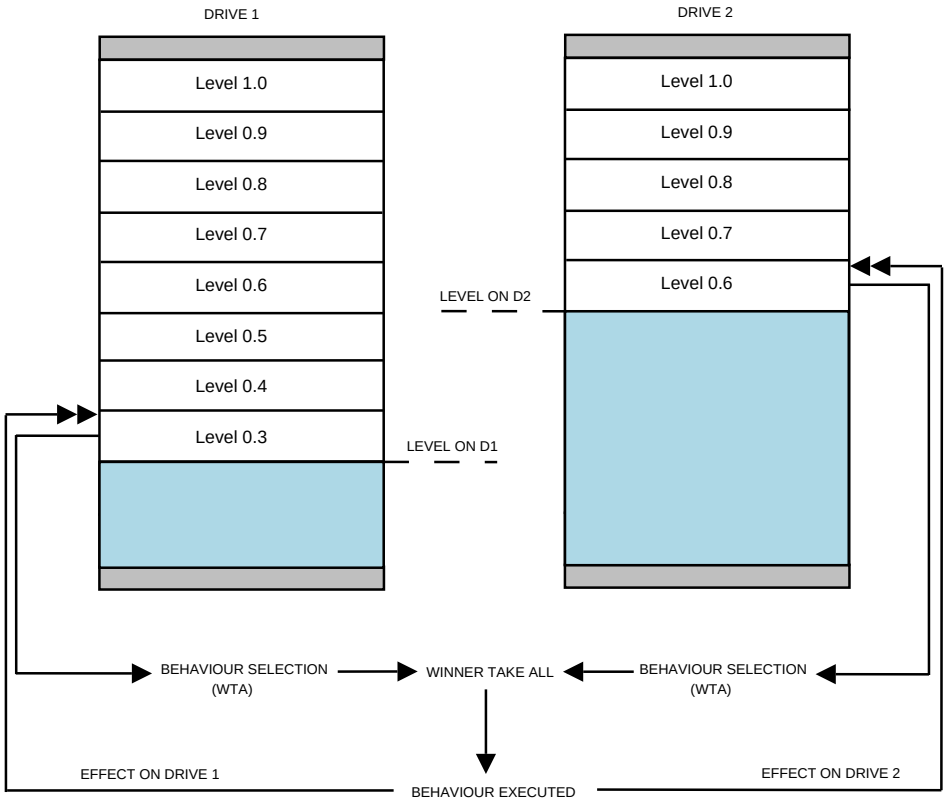


Figure 3.2: Behaviour Selection on 2 Drives

state Drive 2 is in.

3.2.2 Actions

The actions available to the agent are the same regardless of the state its drives are in. Furthermore, for every drive there is the same available set of actions. Since the agent is coupled with the environment, there is no distinction between an action and the effect of that action on the agent. For example, when we describe an action of +0.1 we are really referring to the action that has the effect of increasing the specific drive by 0.1. This action could be “eat” if the drive was energy, or “explore” if the drive was curiosity since it would cause a relative increase in that drive. Our environment model is abstract, hence it makes no difference what type of action it is. It only matters what effect it has on the drives of the agent.

The standard sets of actions that were used in this study can be seen from Tables 3.1 and 3.2. In the case of the agents that were tested with one drive, the available actions for every state are a negative small decrease, a neutral (no effect) action and a positive small increase with the same magnitude as the negative one. For the agents with the two drives all the possible combinations of single-step (i.e. steps of 0.1 magnitude) effects are considered. This was termed a cooperative model since every action is affecting both drives (even the zero change is an action effect).

Action #	1	2	3
Effect on Drive	-0.1	0.0	+0.1

Table 3.1: Single-Drive Agent

Action #	1	2	3	4	5	6	7	8	9
Effect on Drive 1	-0.1	0.0	+0.1	0.0	0.0	+0.1	+0.1	-0.1	-0.1
Effect on Drive 2	0.0	0.0	0.0	-0.1	+0.1	+0.1	-0.1	+0.1	-0.1

Table 3.2: Two-Drives Agent. Double effects

Another setting that was examined for the two-drive agents was to have the same action settings for both drives as for the single-drive model. Every drive had the action setting of Table 3.1 with no action having an effect on the other drive (In this case the "zero effect" on the other drive is not an action effect and it is not used to reinforce the behaviour selection). This encourages the competition between drives and does not allow cooperation (i.e. a behaviour selection that in every step satisfies both drives) since there are no actions that affect both drives. When this set of actions were used the model was termed a competitive one. Finally, other variations on the available actions were tested in the context of an unstable environment which is described further on.

3.3 RL & GA model

The first model tested combines reinforcement learning that accounts for the learning of an agent through its lifetime and a genetic algorithm architecture that provides the evolutionary computation method, necessary for studying the phenomenon of evolution. In this section we analyse each separately in turn and provide the complete model as a consequence.

3.3.1 Reinforcement Learning

Description

In this model a 'full' reinforcement learning (RL) method was used. That is, every action and state transition that the agent undergoes is reinforced according to the perceived reward. Hence, we are dealing with a typical problem of learning from interaction in order to achieve a goal. This satisfies, according to [Sutton and Barto, 2002], the conditions of a 'Reinforcement Learning Problem' and therefore standard update rules were used.

During the life of the agents, a constant balance of exploration and exploitation is used to gain experience of the state-action space. Starting from randomly initialised drive values, the agent is allowed to select actions for a number of steps. This episode is repeated for at least 10,000 cycles in which time the agent has learned the best *possible* behaviour according to the reward received.

Although the state representation is dealing with the internal drives of the agent and not with the environment (usually the states are directly connected with the environment), the state transitions are directly influenced by the effect of the actions which in turn is defined from the environment. At each time step t, the agent receives a representation of its internal drive *state*, $s_t \in S$, where S are the possible states of the drive, and selects an *action* $a_t \in A$, where A are the available actions. At the next time step, the agent receives a numerical reward, $r_{t+1} \in \Re$, and finds itself in a new state s_{t+1} according to the effect that the action had through the environment and on its drives.

TD(0) Learning

All the update rules used within the model belong to the Temporal-Difference Learning methods. This group of methods is using raw experience to learn without having a model for the environment (bootstrapping). The TD(0) method [Sutton, 1988] was used only initially in our model (in conjunction with Q-Learning as action selection mechanism) in order to ensure that the value estimates for the states, $V_t(s_{t+1})$ were converged to the real values $V^{\pi}(s_{t+1})$. The update rule is given by

$$V(s) \longleftarrow V(s) + \alpha[r + \gamma V(s_{t+1}) - V(s_t)] \tag{3.1}$$

where α is the step-size parameter, γ is the discount factor.

According to this rule the value of every state is updated on the basis of the value of the next state and of the reward received by performing that transition, Figure 3.3. However, as we can see from the backup diagram, every transition is dependent on the action that the agent performed and hence the TD(0) rule is used only for the prediction problem (ensuring that the state values were converged) and not for control or action-selection. This is the responsibility of the Q-Learning method.

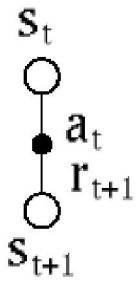


Figure 3.3: Backup diagram for TD(0)

Q-Learning

The Q-Learning algorithm [Watkins, 1989] is a very popular and important method in reinforcement learning. It is an off-policy action selection mechanism in which *"the learned action-value function, Q, directly approximates Q*, the optimal action-value*

function, independent of the policy being followed." - [Sutton and Barto, 2002]. The update rule is given by

$$Q(s_t,a_t) \longleftarrow Q(s_t,a_t) + \alpha\,[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t] \tag{3.2}$$

where again, α is the step-size parameter and γ is the discount factor.

The update rule of the Q-Learning method can be best explained using the backup diagram in Figure 3.4. As was explained in the previous sections, the agent is initialised to random drive states and performs a certain number of steps until the episode is repeated again. Starting from a random state s_t, an action a_t is chosen according to our ε-greedy policy (exploiting 90% and exploring 10% of the time) which leads to state s_{t+1}. From that state we observe the maximum Q-Value and use it as a backup to update the Q-Value for the beginning state and action pair $Q(s_t,a_t)$. This *bootstrapping* technique learns estimates of the state-action values based on other estimates of state-action values.

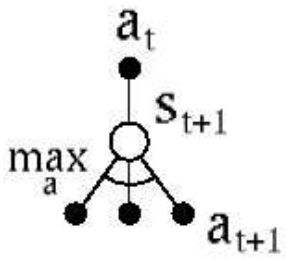


Figure 3.4: Backup diagram for Q-Learning

The Q-Values represent the value of selecting a specific action in a given state. They represent the valency of actions and are those that characterise an agent as successful or not. Since for every drive we have 10 states and in every state the same action set is available, the Q-Value table for describing every state-action will be a matrix of dimensions

$$10\mathbf{x}(\#Actionsperstate * \#Drives) \tag{3.3}$$

For example, a single drive agent with the standard 3 available actions (see Table 3.1) will have a Q-Value matrix of 10**x**3 as can be seen from Table 3.3. The actual values in that table at any moment in time describe the relative preference of the agent to perform certain action over others, when it is in a certain state. The initialisation was always performed by setting those values to 1.

Actions	*-0.1*	*0.0*	*0.1*
STATE	Q-Values		
State 0.1	1	1	1
State 0.2	1	1	1
State 0.3	1	1	1
State 0.4	1	1	1
State 0.5	1	1	1
State 0.6	1	1	1
State 0.7	1	1	1
State 0.8	1	1	1
State 0.9	1	1	1
State 1.0	1	1	1

Table 3.3: Initialising the Q-Values on a 1 Drive agent

Behaviour Selection using Q-Value Tables

Having now described the converged Q-Values, the behaviour selection procedure is further considered. From equation 3.3 we can observe that the size of the Q-Value table is proportional to the number of drives. This is because every drive has its own Q-Values for every available action. Every drive assigns its own valency on the actions available and hence, as it was explained in Figure 3.2, there is a competition between the actions selected from every drive as to which one will be executed overall. Only the selected action's Q-Value is updated on the state that the "winning" drive was in but using the reward from the transitions on both drives.

The method used to select which drive's behaviour will be executed is a Winner-Take-All approach on the basis of the Q-Values magnitude. The behaviour selection on the *single drive* case is the simple solution of that approach. The behaviour with the highest Q-Value is selected and directly executed, affecting only one drive and collecting reward according to the transition on that drive.

Reinforcement Learning Part

The combination of the TD(0) method for ensuring convergence on state-values and the Q-Learning method for action selection, completes our description of the Reinforcement Learning part of the model which can be seen from Figure 3.5. According to the definition of the valency problem (Chapter 2), the behaviour of the agent is unique and developed on the basis of the reward that it receives for every state transition. This reward is part of the agent architecture and it is evolved through the use of a Genetic Algorithm which is the subject of the following section.

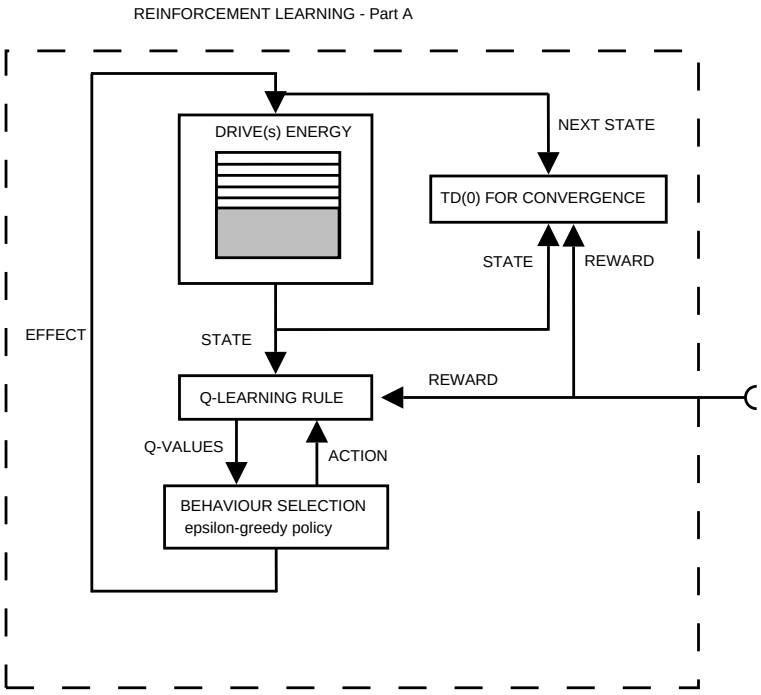


Figure 3.5: Reinforcement Learning Part

3.3.2 Genetic Algorithm Chromosome

Chromosome

As was mentioned before, the agent architecture consists of its drives and its chromosome. The chromosome in this model represents the reward the agent will receive for every state transition occurring on its drive(s). With the use of the genetic algorithm, the chromosome of the agent evolves towards our goal which is to be able to lead the agent to a perfect association between behaviours and their relative value. In other words, the agents will evolve their reward chromosome over generations in order to correctly identify what is good or bad for them. This design allows the study of the valency phenomenon by assigning different chromosomes and examining different emerging behaviours. These range from the extremes of an agent that correctly identifies all the right behaviours (with respect to its drives) as good ones, to an agent that identifies wrong behaviours as good ones.

Since the chromosome is the reward for state transitions, then the length of the chromosome is a function of the number of states per drive (S_d) and the number of drives (D). This function is

$$Length = S_d^2 * D \tag{3.4}$$

and depends on the square of the number of states per drive which is the number of possible transitions per drive. For example, on an agent with 1 Drive its chromosome is a 100 sequence, whereas for a 2 Drive agent it is a 200 sequence. As we can see, the length increases linearly with respect to the number of drives. This was achieved, as explained before (see Figure 3.2), by considering a behaviour selection per drive which is independent of the state of the other drive.

The reward values used for the chromosomes were integers within the range of 0 to 10 inclusive. In Figure 3.4 a typical chromosome for a 1-Drive agent is shown in relation to the actual interpretation as reward values for state transitions. The chromosome of the agent is the 100 reward values that the agent can receive when performing a certain transition. For instance, if from a drive state of 0.1 the agent manages to reach a drive state of 0.2, by choosing an appropriate action, it will receive a reward of 9.

	Ending State s_{t+1}										
Starting State s_t	*STATES*	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*	*1.0*
	0.1	2	9	1	1	0	8	2	3	10	3
	0.2	4	7	0	0	8	1	8	1	9	3
	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.
	1.0	2	.	.	.	.	.	.	.	.	10

Table 3.4: Example of Reward Chromosome for a 1-Drive Agent

3.3.3 The Complete Model

Having described the reinforcement learning cycle and defined what the chromosome represents, we can now describe the complete model. This architecture is able to evolve reward values that can be used by the Q-learning algorithm in order to produce different behaviours. According to the fitness function in the competition, we can lead our model to find the appropriate reward values that can produce successful behaviours and correct associations between actions and valency of actions.

As can be seen from Figure 3.6, initially a population of chromosomes with random numbers $\in [0, 10]$ is produced in the Random Initial Pool. Each of these chromosomes is an agent's reward function for state transitions that is used in the RL part of the model. After at least 10,000 cycles per chromosome of learning, the agent has developed its Q-Values that govern the action-selection behaviour (based on the valency of actions) based on that particular reward chromosome. The Q-Values are then used by the agent to select actions and it is scored on a competition test; according to its fitness the chromosome (reward values) then goes through the standard GA part of selection, crossover and mutation. The procedure repeats for G generations

The only ad-hoc in the GA part of the model is the 1% probability of the chromosome to be inverted. This inversion was included in order to help variability within the population and also to provide the opportunity of changing "local order". By this term we characterise the special attribute of our chromosome which is that local relations between rewards (e.g. neighbouring alleles) can greatly influence the action-selection

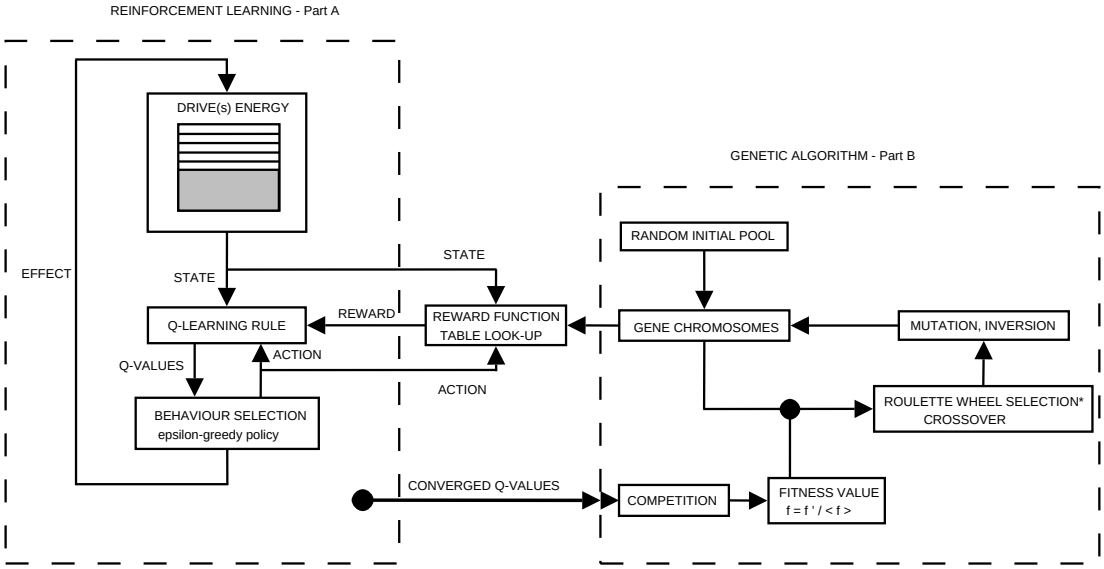


Figure 3.6: The RL & GA Model

behaviour. Thus, inversion changes local order and helps to avoid premature convergence of the chromosome pool.

For example, consider the 1st and 2nd allele of our chromosome. Those are reward values for state transitions from a drive level of 0.1 (because of the way we map the chromosome onto a 10x10 matrix, see Figure 3.4). The 1st one is the reward for zero transition and the 2nd for a transition to a drive level of 0.2. If the respecting rewards are 1 and 10, then the transition that gets the 10 reward will *probably* be preferred -our agents are “far-sighted” so future expected reward could change that. Hence, by inverting that local relationship a very different behaviour could emerge.

That particular phenomenon is also the main reason why two variants of the model were examined. The only difference between them was the use or not of the crossover mechanism. The idea behind that is that since small local changes can have such a great effect, then perhaps only mutation can lead towards evolving successful agents. This was further supported based on the work of [Mclean, 2001] where again only mutation on the weights of a neural network was used to evolve successful agents. Besides, in a neural network it is not only the case of a “local order” phenomenon but also due to

general relational effects.

Competition

In order to score the agents and validate their fitness a competition was performed at the end of the reinforcement learning cycles. The converged Q-Values were used in order to examine the action-selection mechanism that the agent developed. Starting from the minimum level of 0.1 on the drives, the agent was examined for a number of steps and scored based on its overall performance. The target is to reach the goal of 1.0 in all drives (satisfy its internal drives) and to remain in that level. Depending on the experiment and on the specific problem, different fitness functions were used to guide evolution. The fitness functions that were used are presented in Table 3.5

Number of Drives	**Competition Steps**	**Fitness Function**
1	11	A*Max+B*10/Steps+C*Final
2	21	A*(Max1*Max2)+B*(Final1*Final2)
2	21	A*Max(c)+B*20/Steps(c)+C*Final(c)

Table 3.5: Typical Fitness Functions

where *A, B, C* are parameters $\in$ [1,10] , *Max* is the maximum level on the drive that the agent can reach throughout the competition, *Steps* are the number of steps needed for that maximum to be reached and *Final* is the final level the drive(s) reach. The c parameter refers to a situation where the competition was split into 2 sub-competitions each one using the Q-values from only one drive at a time. This necessity will become apparent from the experimental results.

In Table 3.6 the first model is summarised in an algorithmic fashion. Table 3.7 presents the main parameters of the model.

```
Initiate random population of Chromosomes
For G Generations:
   For every Chromosome:
      While Q-values not converged
         Initialise Drive level randomly
         For N steps:
            Take action, observe reward and new state
            Q-learning update
         End
      End
      Compete with QValues and score Chromosome
   End
   Selection, Crossover, Mutation
   New Chromosomes
End
```

Table 3.6: The RL & GA model in Algorithmic format

Parameter	**Value(s)**
epsilon ε	0.1
gamma γ	0.9 (0.2, 0.6, 0.99 tested)
alpha α	0.1
Mutation rate	1% & 8%
Inversion rate	1%
Reward Range	[0,10]
Q-Values Initialisation	1

Table 3.7: Typical parameter values of the 1st model

3.4 Q-Evolution model

The second model that was used in this study is based entirely on an evolutionary mechanism in order to develop successful valency associations. There is no learning involved in the form of a separate mechanism as we saw for the RL & GA model. The only way of succeeding towards correct action-selection is by following the standard genetic algorithm steps of selection, crossover and mutation. We evolve directly the Q-Values that are used for the behaviour exerted by our agent and are the measure of the valency of actions.

The chromosome of the agent in this model is the actual Q-Values needed to guide the action-selection mechanism. Instead of evolving reward values to be used for the creation of Q-Values we are directly evolving the last. Two different ranges of values were used for the chromosome, a small range [1,10] and a normal one [1,100] that was similar to the one obtained from the first model results. Also the length of the chromosome is now reduced as it does not need to describe every state transition on the drive(s) of the agent but only to account for the necessary number of Q-Values. The function describing the length of the chromosome on this occasion is:

$$Length = S_d * A_d * D \tag{3.5}$$

where S_d is the number of states per drive (10 throughout our study), A_d is the number of actions available per drive, and D is the number of drives. For instance, an agent with two drives and the cooperative set of actions (9 actions per drive, see Table 3.2) will have a chromosome of 180 length which is by 20 shorter than the analogous RL&GA model chromosome.

The model is presented in Figure 3.7. As it can be seen, it is basically the RL&GA model without the learning part of the model. This time we are directly evolving the Q-Values that are used in the competition test to perform the action selection of the agent by simply selecting the action with the maximum Q-Value per state. The competition and the rest of the genetic algorithm mechanism are exactly the same as the first model with the only difference being that this time the chromosome of the agent is not the reward but the Q-Values. This model represents a strictly evolutionary scenario and it is used extensively for comparison reasons in order to discover the benefit or cost of

learning in our models.

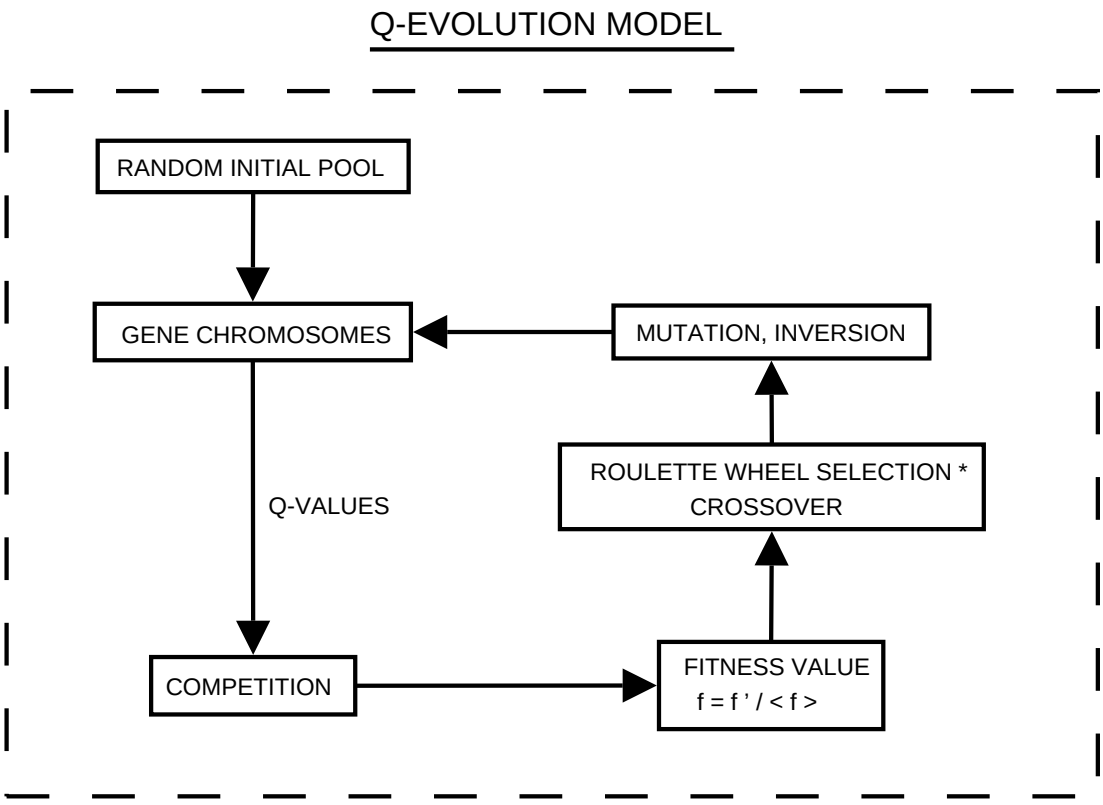


Figure 3.7: Q-Evolution Model

An algorithmic format of this model is given in Figure 3.8 and some basic parameters are summarised in Table 3.9.

```
Initiate random population of Chromosomes
For G Generations:
   For every Chromosome:
      Compete with QValues and score Chromosome
   End
   Selection, Crossover, Mutation
   New Chromosomes
End
```

Table 3.8: The Q-Evolution model in Algorithmic format

Parameter	**Value(s)**
Mutation rate	1% & 8%
Inversion rate	1%
Q-Values range	[1,10] & [1,100] (default)

Table 3.9: Typical parameter values of the 2nd model

3.5 RL & GA model with a NN for reward

The third and final model is another variant of the first basic model. The main difference this time is that we are not evolving the reward values but the synaptic weights of a neural network that provides the reward. This was done in order to tackle the problem of chromosomatic length growth, as the number of drives, states per drive, actions and size of stochasticity increases. The length of the chromosome is an important parameter both for the processing speed of the algorithm and the viability of our assumptions that this method can be generalised over more complex behaviours.

The use of a neural network by evolving its synaptic weights in order to improve agent performance over a task has been a very common practise these last years, [Batali and Grundy, 1996, Harvey, 1996, Mclean, 2001, Nolfi, 1994, Suzuki and Arita, 2003] and others. By using a simple neural network architecture we can greatly reduce the length of the chromosome since we only need to evolve the synaptic weights as it can be seen from the model diagram in Figure 3.8.

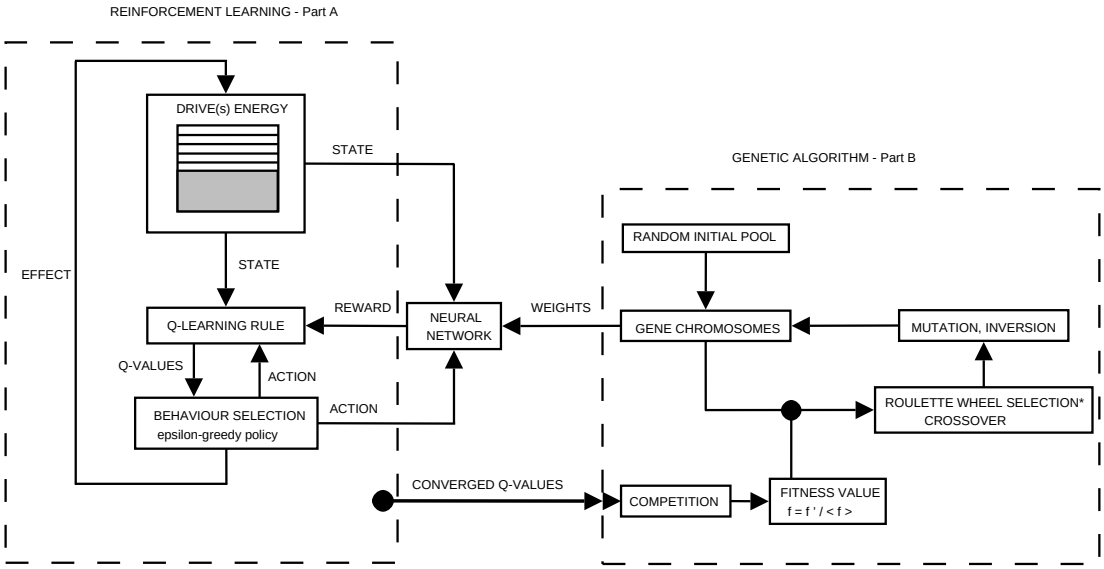


Figure 3.8: RL & GA with NN Model

Our model includes the same reinforcement learning and genetic algorithm cycles as the first model but now a neural network, see Figure 3.9, receives the information

necessary from both cycles in order to compute the reward. In detail, the weights of the neural network are provided from the genetic algorithm in the form of the agent's chromosome. Having now a defined architecture for the network, all possible state transitions and actions are passed to the network as inputs[1]. The network based on those inputs and the synaptic weights, is using a simple summation rule over the inner product of inputs and weights per node, equation 3.6, and then a sigmoid function, equation 3.7, in the two layers (hidden and output layer) in order to compute the output of the network for every set of inputs.

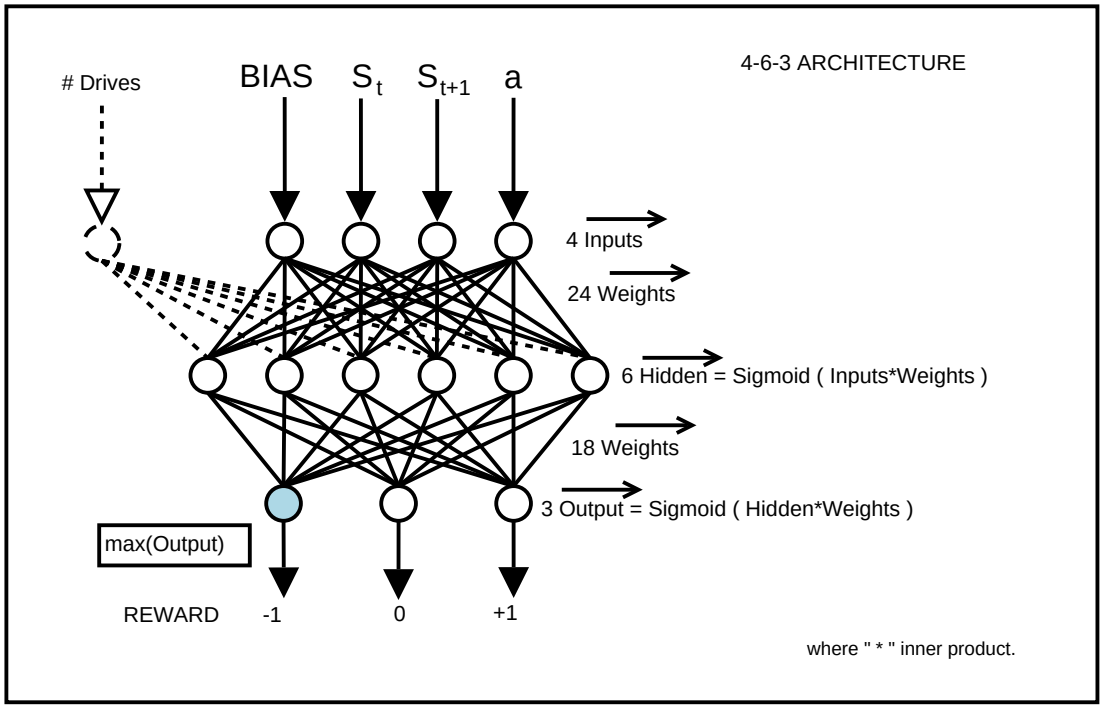


Figure 3.9: The Neural Network

Analytically, consider a set of inputs X_i and a set of 1st layer weights W_{ij}, $i \in [1,4], j \in [1,6]$. Then the output of the hidden layer for every node j will be

$$H_j = \sigma(\sum_{i=1}^{4} W_{ij} X_i) \tag{3.6}$$

[1]The neural network is used only once per agent. Instead of passing every state transition and action selected as inputs to the network in real time, we use the network once to calculate all the possible rewards and we store them to be used directly in real-time.

where σ is the sigmoid function given by

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3.7}$$

Following again the same procedure of summing the inner product and passing it through the sigmoid function, the final output of the network for every node z $\in [1,3]$ is given by

$$O_z = \sigma(\sum_{z=1}^{3} W_{j,z} H_j) \tag{3.8}$$

As can be seen from the model diagram, Figure 3.8, the output of the three final nodes is compared and the most activated node is selected. The three output nodes represent three different numerical rewards, namely -1, 0 and +1. Hence by selecting the most activated node we select the reward for the particular input values of the network. In that way a reward value, out of the three possible, is selected according to the state transition and action.

The only difference between that network which is designed for a single drive agent and the one used for two drives is an extra input node added on the multiple drive case (symbolised with dashed lines in Figure 3.8). That unit simply represents numerically the number of drives of the agent and plays an important role in the variation of reward given to every drive. Without that, identical state transitions (due to identical actions) on different drives would cause the same reward. This is because the input would be exactly the same. This extra node provides variation on the reward that the agent will receive for the same action and state transitions but on different drive.

This model attempts to tackle a major problem of dimensionality. Our two previous models increase in complexity as the number of drives for the agent increases, because of the fact that the length of the chromosome is a function of that number (Eq. 3.4 and 3.5). This is not the case for the third model since with only one extra input node over the single drive architecture, the network can account for multi-drive agents. That configuration though cannot approximate *any* reward function.

The genetic methods on the synaptic weights were performed initially as in the other models, testing the model with both crossover and mutation and only mutation. A similar technique to [Mclean, 2001] was also examined by performing crossover between set of weights that belonged to one node, instead of random-point crossover.

This was done on the basis that local information (i.e. the weights of one node) is important and the performance of a node is dependent on the interaction of those weights. Hence, we treat those weights as a whole and instead of crossing over chromosomes in random points we cross them on points where groups of weights start or end.

Different network architectures with variations on the hidden layer size were also examined for performance evaluation reasons. A neural network without a hidden layer was used and also a hidden layer with only 2 hidden units. Finally, the range of the synaptic weights examined was from -1 to +1 with increments of 0.1 and the bias was set always to 1. The algorithmic format of the model and the typical parameters can be seen from Table 3.10 and Table 3.11 respectively.

```
Initiate random population of Chromosomes
For G Generations:
   For every Chromosome:
   Create the NN from the Chromosome (Synaptic Weights)
   From the NN create the rewards for all transitions
      While Q-values not converged
         Initialise Drive level randomly
         For N steps:
            Take action, observe new state and reward from NN
            Q-learning update
         End
      End
      Compete with QValues and score Chromosome
   End
   Selection, (Crossover), Mutation
   New Chromosomes
End
```

Table 3.10: The RL & GA model with a NN in Algorithmic format

Parameter	**Value(s)**
Mutation rate	1% & 8%
Inversion rate	1%
Rewards	[-1 0 +1]
Weights Range	[-1.0:0.1:1.0]
Hidden Units	0, 2, 6 (default)
Bias	1.0

Table 3.11: Typical parameter values of the 3rd model

3.6 Dynamic Environments

The agent-environment interaction so far has been described as deterministic and stable. Deterministic in the sense that actions lead to certain effects with a probability of 1 because the environment is not introducing any stochasticity and stable in the sense that the environment is not changing over time (time as in number of generations). Both those assumptions were challenged in our study by introducing two important aspects in the model: energy decay and an unstable environment.

3.6.1 Energy Decay

According to our specification of the agent drive(s), there is a need for the agent to ensure that its drives become as "satisfied" as possible by performing various actions on different states. Since every action has a direct effect on the drives it was just a matter of choosing the right actions. However, actions have no cost and they lead to the same effect always. The energy decay function tackles both those aspects as it introduces a decay on the level of the drives which is given by

$$Energy_{t+1} = Energy_t * 0.9 \tag{3.9}$$

according to this the level of the drive after each step is

$$Level_{t+1} = Level_t * 0.9 + < effect\ of\ action > \tag{3.10}$$

This function introduces an exponential decay on the level of each drive which is independent of the agent's action selection mechanism (i.e. the decay is always present). It was termed energy decay because the drives resemble energy accumulation in the form of nutrition or perhaps charging for a robot. Since each drive has 10 discrete levels of 0.1 magnitude, the decay was recorded until it reached a difference of at least 0.1 from the current level perceived by the agent. Then, the sudden reduction on the drive was implemented.

For example, a simple scenario is considered in Table 3.12. Starting from a level of 0.1 for the drive, 5 steps are described with various actions being selected each time. The perceived level is the level that the agent can notice and the actual one is the "real"

Action performed (effect)	**Perceived level**	**Actual Level**
—(Start)—	0.1	0.1
+0.1	0.2	0.1*0.9+0.1=0.19
+0.1	0.3	0.19*0.9+0.1=0.271
0	0.3	0.271*0.9+0=0.244
0	0.3	0.244*0.9+0=0.219
0	0.3 reduced to 0.2	0.219*0.9+0=0.19

Table 3.12: Example of the energy decay function

level. The energy decay occurs before the effect of the chosen action is added on the current level of the drive, as described in 3.10. Notice how on the last step, although the agent chooses an action with an effect of 0 on its drive the actual level reaches the 0.1 difference limit with the perceived level and hence the latter reduces from 0.3 to 0.2.

3.6.2 Unstable Environment

The final aspect of our model architecture is the implementation and use of an unstable environment in which a sudden change occurs after a certain number of generations. This implementation is similar to the one from [Mclean, 2001] in which two different objects, food and poison, flipped properties after certain number of generations. Our approach follows that flipping behaviour by changing the effect of actions every N Generations. Hence, a previously reinforced action that had a positive effect on the drive will be flipped to have exactly the opposite effect.

The unstable environment accounts for sudden changes occurring in the environment during the course of evolution. By posing those changes we investigate the adaptability of our models and the effect and cost of learning throughout the life of the agents. This environment provides another perspective on our models and tests their ability to respond to changes after generations of agents have been evolved on a completely “opposite” environment. Only the models that are extremely adaptable will be able to deal with such instabilities, especially when the environment keeps on

changing from one generation to the other.

The intervals between generations for a change to occur were kept the same as those of [Mclean, 2001], see Table 3.13, and the change of action effects was to their direct opposites except when the effect was zero, when it remained the same, as can be seen from Tables 3.14 and 3.15. The flipping pattern was kept constant changing effects of actions every N generations.

Generations
50
25
13
7
1

Table 3.13: Number of Generations after which a change occurred

Action #	1	2	3
Effect on Drive	-0.1	0.0	+0.1
Effect on Drive	+0.1	0.0	-0.1

Table 3.14: The two different effects on an unstable environment for a single-drive agent

Action #	1	2	3	4	5	6	7	8	9
Effect on Drive 1	-0.1	0.0	+0.1	0.0	0.0	+0.1	+0.1	-0.1	-0.1
Effect on Drive 2	0.0	0.0	0.0	-0.1	+0.1	+0.1	-0.1	+0.1	-0.1
The opposite effects									
Effect on Drive 1	+0.1	0.0	-0.1	0.0	0.0	-0.1	-0.1	+0.1	+0.1
Effect on Drive 2	0.0	0.0	0.0	+0.1	-0.1	-0.1	+0.1	-0.1	+0.1

Table 3.15: The two different effects on an unstable environment for a 2-drive agent

Chapter 4

Results

In this chapter we present the results of our simulated experiments for the three models introduced in the previous chapter. The order of appearance follows the structure of the previous chapter, firstly the “RL and GA” model, secondly the “Q-Evolution” model and thirdly the “RL and GA with NN” model. Finally, the results for unstable environments close the presentation of results.

Every model was tested on four different cases with one or two drives and with or without energy decay. These four cases are presented with an emphasis on the resulting differences between them. Furthermore, another subdivision of the results is caused from different mutation rates, crossover mechanisms and different fitness functions applied. In order to present the results coherently and compactly, those extra subdivisions are incorporated into the four different cases.

The parameter space of the models is large due to the nature of the genetic algorithm and the reinforcement learning rules. In many cases the exploration of that space was restricted for time consuming reasons since it was judged to be out of the goals of this study. However, when necessary this exploration was performed by adjusting the models on different parameter settings and comparing their behaviour.

Typical times needed to produce the results for the “RL & GA” and “RL & GA with NN” models (multiple solutions per run) are around 10 hours for single drive cases (30 Gen.), 15-20 hours with energy decay (50 Gen.), 45-50 hours for two drives (100 Gen.) and 75-80 hours for two drives with energy decay (100 Gen.). The Q-Evolution model needed less time for all cases with a maximum of 10 hours for a two-drive run with

energy decay, averaged over 50 trials and for 50,000 generations.

4.1 First Model - RL and GA

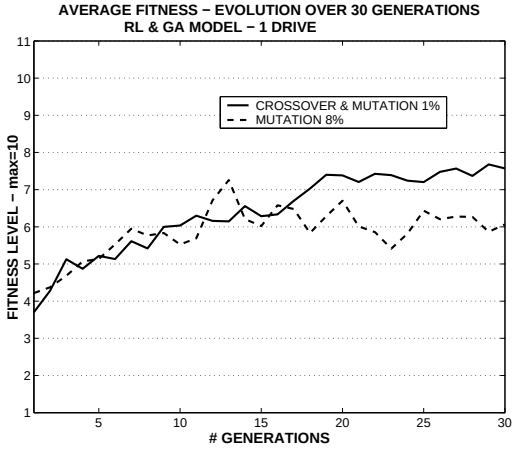
This section reports the results of the first model which addresses the combination of evolution and learning on the valency problem by directly evolving the reward function. The results presented concentrate on the maximum and average fitness development over generations and on characteristic performances over the competition stage. Also, an example of the actual evolved chromosome of reward is given together with the converged Q-Values emerged due to that chromosome. Results produced by using only mutation in the GA cycle are presented and compared with cases using both mutation and crossover.

Throughout this section generality is an issue since our results for this model are not averaged over many runs due to computational and time limitations. However every result presented is typical of its kind and it was not "chosen" for performance reasons.

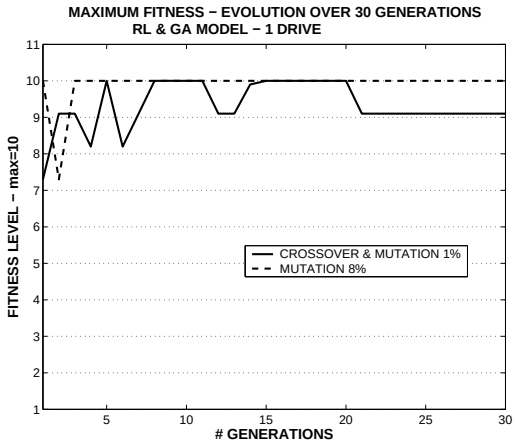
4.1.1 One Drive

We report results for the single drive case with a population of 50 agents per generation. In Figures 4.1(a) and 4.1(b) the average and maximum fitness of the population are presented. Two different settings are compared, using only mutation with a rate of 8% and using both crossover and mutation (standard GA) but with a mutation rate of 1% only. As can be observed, the standard GA setting achieves a higher average fitness in the population than by using only mutation. However, in the maximum fitness performance, the standard GA setting flips between the perfect solution (fitness level of 10) and another high solution (fitness level of 9.1), whereas the only-mutation setting continuously achieves a perfect solution.

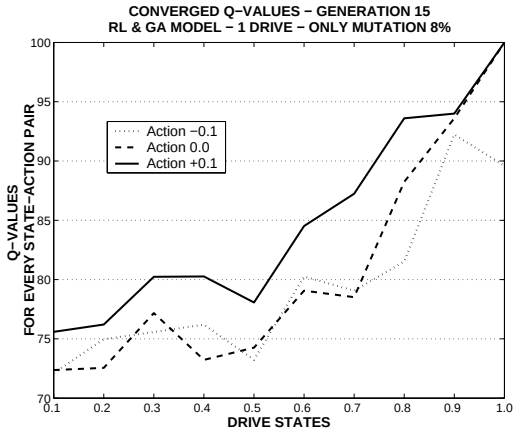
Figures 4.1(c) and 4.1(d) show the converged Q-Values from the agents achieving maximum fitness for both settings. For both cases the highest Q-Value (valency of actions according to state) for almost every state refers to the action with the +0.1 effect. Figure 4.1(e) shows a "perfect" competition performance and as it can be seen,



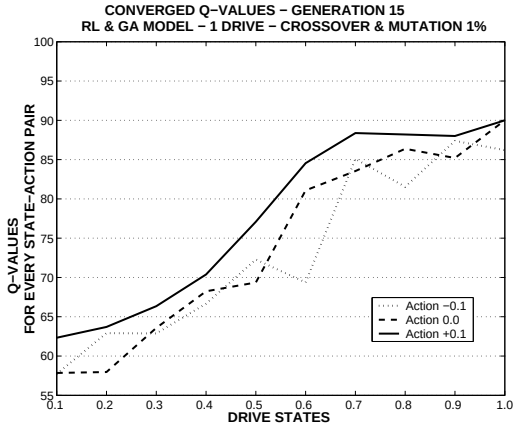
(a) Average Fitness



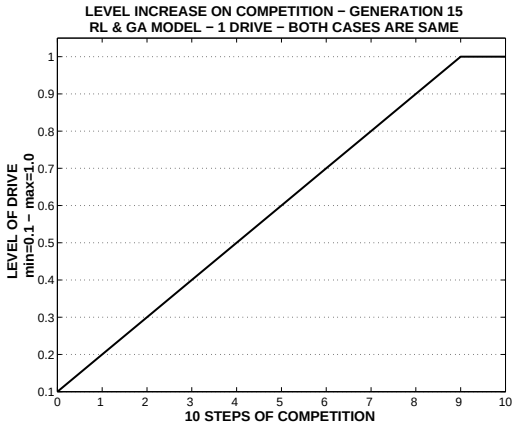
(b) Maximum Fitness



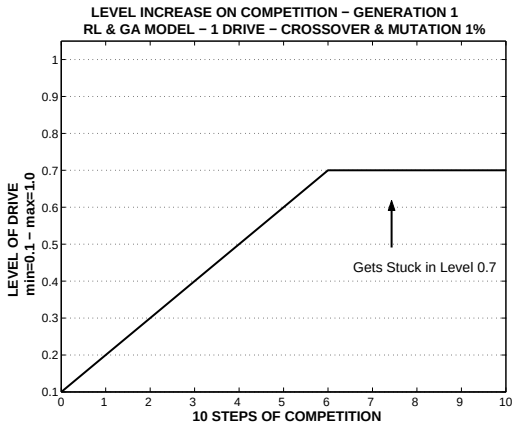
(c) Only Mutation of 8%



(d) Crossover and Mutation 1%



(e) Best Performance



(f) Medium Performance

Figure 4.1: RL & GA Model with 1 Drive

only in the last state does the action +0.1 have the same Q-value as the action 0.0. This is because in the last step of the competition both those actions have the same effect due to the hard bounds of the drive. Both actions will cause the level of the drive to remain at the top level of 1.0. Figure 4.1(f) displays an unsuccessful agent that gets stuck in level 0.7 and remains there always choosing action 0.0.

The evolved reward function that a typical successful agent is using can be seen from Table 4.1. Only the rewards that can be used from the agent in this case are displayed (e.g there is no action that can lead the agent from level 0.1 to level 0.7, no action with an effect greater than +0.1 or less than -0.1) and also the mean reward that the agent can collect from a beginning state. As can be seen there is an increase in the mean expected reward towards the actual goal of evolution which is to evolve agents able to reach state 1.0.

	Ending State s_{t+1}											
Starting State s_t	State	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	**MEAN**
	0.1	3	5									**4**
	0.2	7	1	4								**4**
	0.3		6	4	3							**4.3**
	0.4			7	5	1						**4.3**
	0.5				9	0	1					**3.3**
	0.6					0	5	5				**3.3**
	0.7						9	4	9			**7.3**
	0.8							2	7	9		**6**
	0.9								8	6	7	**7**
	1.0									7	9	**8**

Table 4.1: Example of a successful reward function

Our Q-Learning algorithm is“far-sighted” and using that evolved reward function leads the agent to correct valency associations and hence to those future states of high reward. In many cases we can observe that the *local* reward relations should lead the agent to “wrong” states, away from the goal state of 1.0. For example starting from state 0.5 a reward of 9 is given if the agent moves to state 0.4 (hence choosing an action

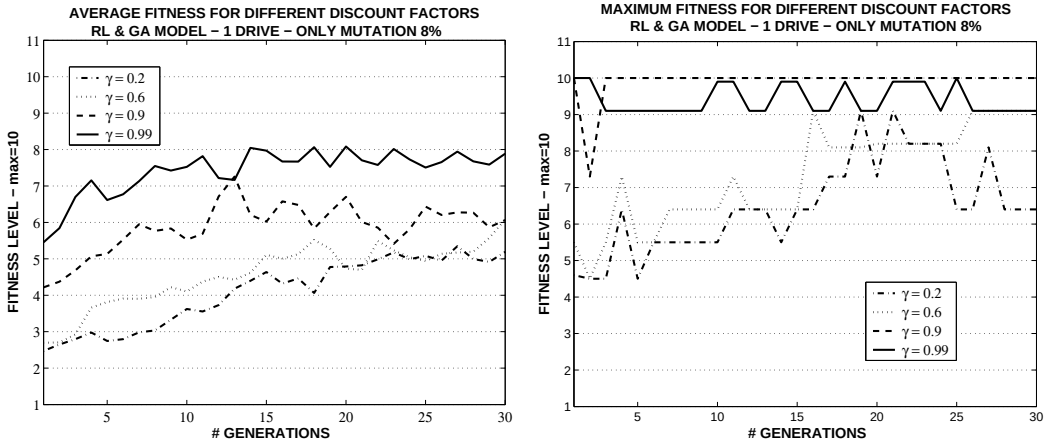
with -0.1 effect) and only a reward of 1 if it moves to state 0.6. However, due to the nature and discount rate setting of our Q-Learning cycle and the fact that there is also exploration the agent is able to evaluate expected future reward and that is why this evolved reward function leads to a successful agent that is able to achieve satiation on its drives.

4.1.2 Exploration of γ values

The discount factor setting has a great influence on the performance of our model. It determines the present value of future rewards and hence specifies the "sight" of our agents. Four different settings were compared on the single drive model and the results are presented in Figures 4.2(a) and 4.2(b). Based on the results the setting chosen for the model in subsequent experiments was a discount factor of 0.9 since although it provides a second-best average fitness for the population, it is more stable in achieving a highest fitness agent.

The smaller the discount factor is, the worst the average and maximum fitness performance becomes. This can be explained from the fact that small discount rate implies a "myopic" agent that is concerned only with maximising immediate rewards. Such an agent can only be successful in the competition if its reward function is "guiding" it in *every* state and hence the evolution of an appropriate reward function becomes harder. It is also arguable if this would be biologically plausible.

The maximum discount factor tested was $\gamma = 0.99$ and although it produced a higher average fitness, it did not manage to outperform our chosen setting of $\gamma = 0.9$ for the maximum fitness per generation (c.f. 4.3(a)). The results for that case indicate that the maximum fitness is fluctuating between two high fitness values of 9.1 and 10. That is an effect of the agents being too "far-sighted" and not managing to remain in the goal state because they do not "care" as much about immediate reward.



(a) Average Fitness (b) Maximum Fitness

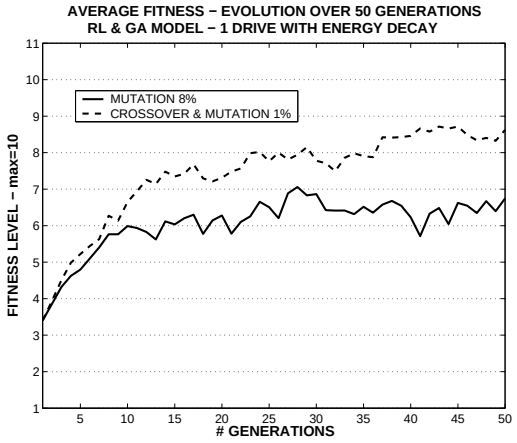
Figure 4.2: Comparison of different γ value settings

4.1.3 One Drive with Energy Decay

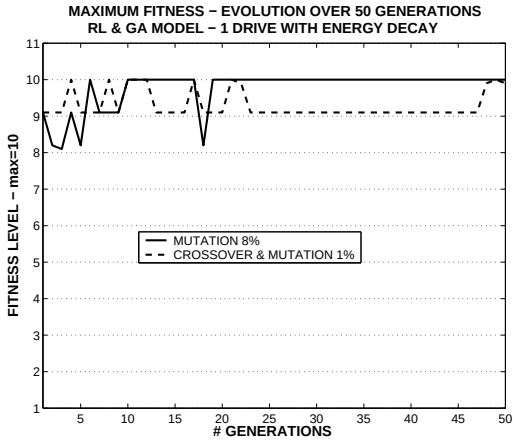
The case of the single drive but with energy decay added is presented in this section. In Figures 4.3(a) and 4.3(b) the average and maximum fitness is plotted for two different GA settings. The same effect can be observed here as with the previous case. The standard GA setting of both crossover and mutation performs better than the only-mutation setting for the average fitness of the population but it flips between two good solutions at the maximum fitness.

The energy decay function has the effect of separating the converged Q-Values in the last state as it can be seen from Figures 4.3(c) and 4.3(d). In order to achieve maximum fitness the agent has to remain on the final state of 1.0. The Q-Values (valency) of actions +0.1 and 0.0 separate since now the action 0.0 will not always lead the agent to remain in that state because of the energy decay. Hence, the effect of adding the energy decay function is that successful agents will be constantly pursuing action +0.1 even when they are in the desired state. They become in a sense greedy.

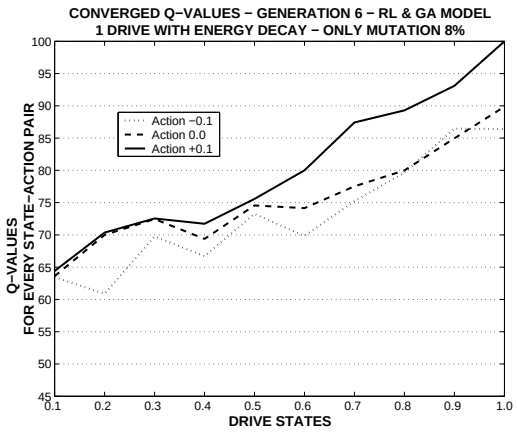
The energy decay is applied only in the learning cycle, not in the competition. In the competition the energy decay was only recorded and not used to drop the agent in a lower state. This was done for computational reasons; if the decay was applied in



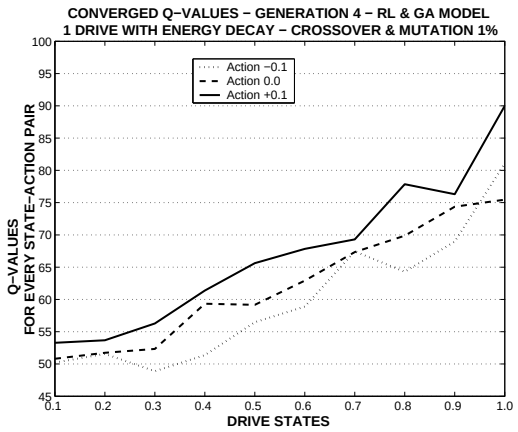
(a) Average Fitness



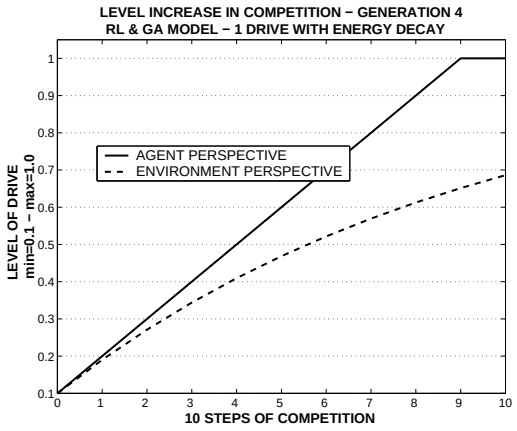
(b) Maximum Fitness



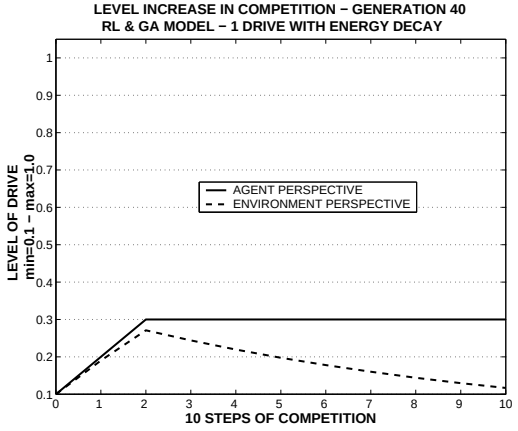
(c) Only Mutation of 8%



(d) Crossover and Mutation 1%



(e) Best Performance



(f) Bad Performance

Figure 4.3: RL & GA Model with 1 Drive and Energy Decay

the competition, the agent would need more steps, hence more time, to reach the goal state. Figure 4.3(e) displays the competition performance of a successful agent that reaches and remains at state 1.0 and Figure 4.3(f) displays an unsuccessful agent that gets stuck in level 0.3. The "Agent Perspective" is the information on the level of the drives that the agent receives and what was used to score its performance. In contrast the "Environment Perspective" is the actual development of the energy level, including the energy decay, that the agent cannot perceive because of the discretisation of its states.

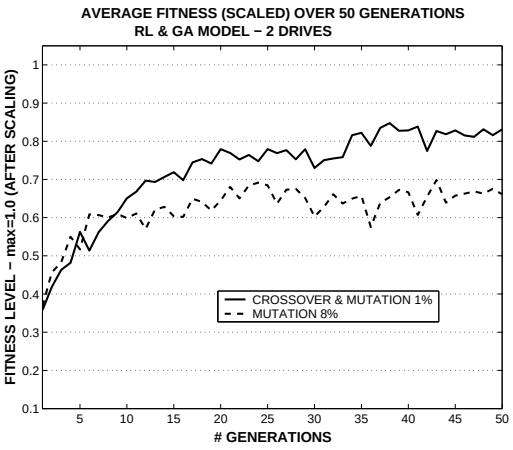
4.1.4 Two Drives

The results for the addition of a second drive on the RL and GA model are presented in this section. The size of the population was increased to 100 agents per generation and the size of the reward chromosome increased to 200 according to Equation 3.4, as was described in Chapter 3.

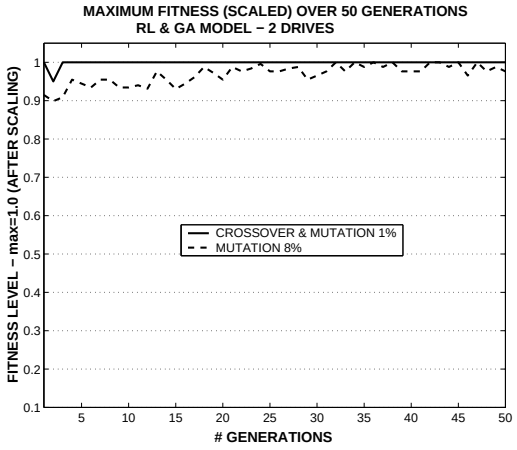
In Figures 4.4(a) and 4.4(b) the average and maximum fitness over 50 generations is presented. As opposed to the single drive case, the standard GA setting of crossover and mutation with a mutation rate of 1% performs better even in the maximum fitness. The only-mutation setting is unstable due to the high rate of 8% mutation and needs more generations to reach the perfect solution. The fitness level is scaled to 1 since different parameters on the fitness function were used for each model.

In Figure 4.4(c) one emerged behaviour can be observed from the converged Q-Values. Every drive develops different Q-Values for the 9 possible actions with drive B having control over drive A by developing greater Q-values for all actions. From all the actions, the largest Q-Value for all states is attributed to the 6^{th} action (see Table 3.2) which corresponds to a positive increment of +0.1 to both drives. Drive's A selected behaviours are never used since drive B has developed higher valency for all actions in every state and has the highest Q-Values for a behaviour that satisfies both drives. As can be seen from Figure 4.4(d) that behaviour is very fast and efficient in leading both drive levels to the goal state of 1.0.

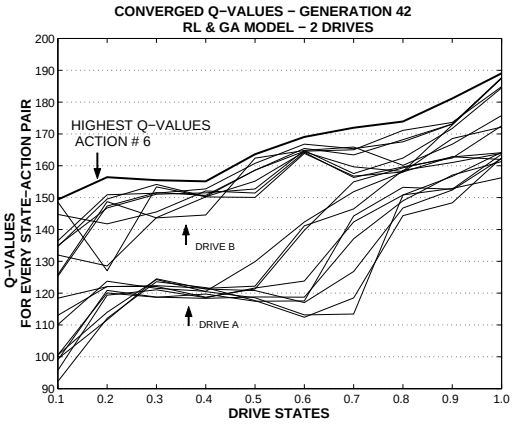
Another completely different emerged behaviour is presented in Figure 4.4(e). The converged Q-Values for both drives are varying and different actions develop the



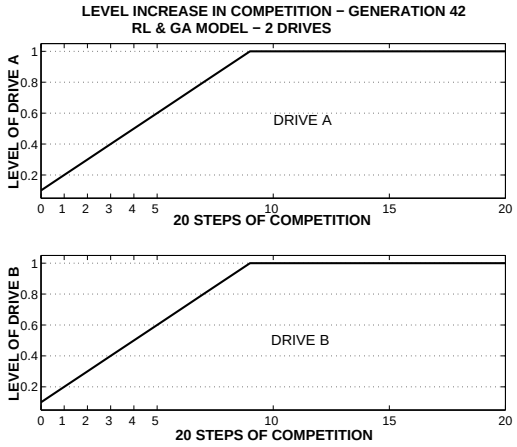
(a) Average Fitness



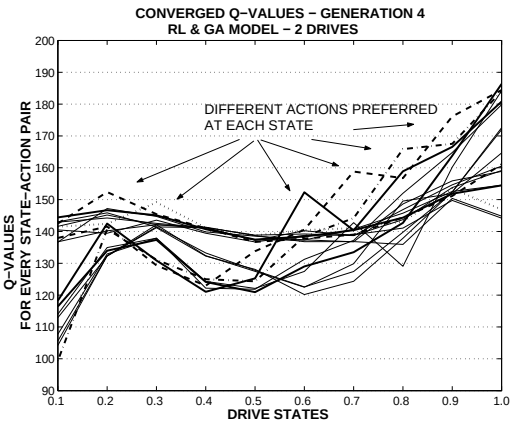
(b) Maximum Fitness



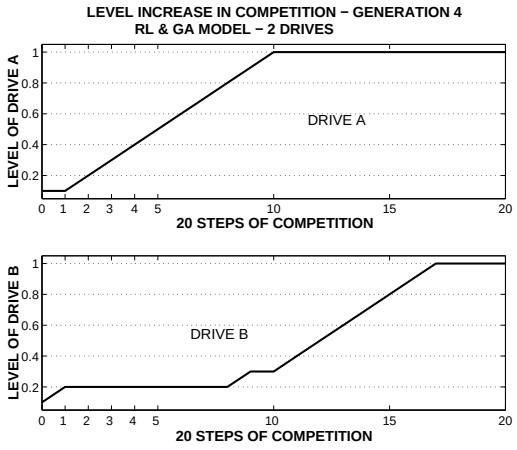
(c) Converged Q-Values



(d) Competition Performance



(e) Converged Q-values



(f) Competition Performance

Figure 4.4: RL & GA Model with 2 Drives

largest valency per state. That leads to an interchanging behaviour execution where both drives participate and cooperate in order to reach the goal state of 1.0. As it can be seen from Figure 4.4(f) this behaviour needs more steps to reach the target but in a way it is more interesting because it creates "paths" among behaviours instead of just choosing a single behaviour that is the most efficient.

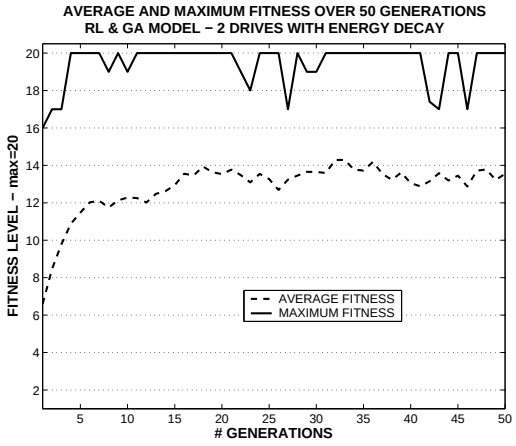
This type of behaviour converges faster and is the typical behaviour observed in the majority of these results. This is explained from the fact that it is easier to evolve such an interchanging behaviour that receives a maximum fitness score. In contrast, the first type, which only selects the best possible action for both drives, is a solution represented by a singularity within the fitness landscape and hence difficult to be located since its fitness is equal to an "easy-found" solution. Due to that fact, the second type of behaviour is not disappearing after the 42^{nd} generation and it is the dominant one. In order to produce a dominant first type of behaviour, the fitness function has to be increased in difficulty level and apply extra pressure on the population.

4.1.5 Two Drives with Energy Decay

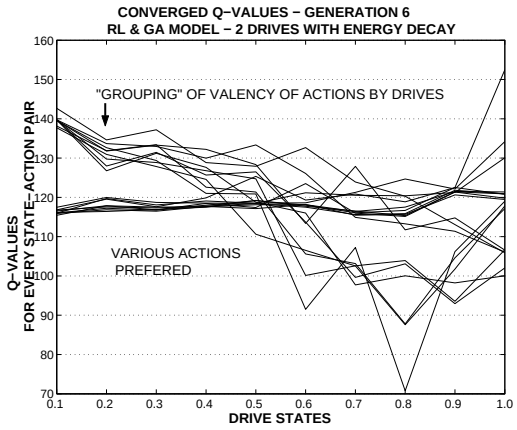
The inclusion of the energy decay function on the two drive case and its effect are considered here. In Figure 4.5(a) the average and maximum fitness of the population are presented for the standard GA setting of crossover and mutation. The only-mutation setting was not examined, since in the previous case without the energy decay proved to be the worst for both average and maximum fitness.

The increase in the fitness is rapid once more and a perfect solution is achieved continuously in many generations. The fitness function used in that case (see Table 3.5) is not applying extra selective pressure to reach the target of 1.0 level within a small number of steps but instead it only examines the maximum level reached and the level the drives are in at the last competition step. This encourages various behaviours and combinations of them to be eligible for maximum fitness even if their effect is not the best possible.

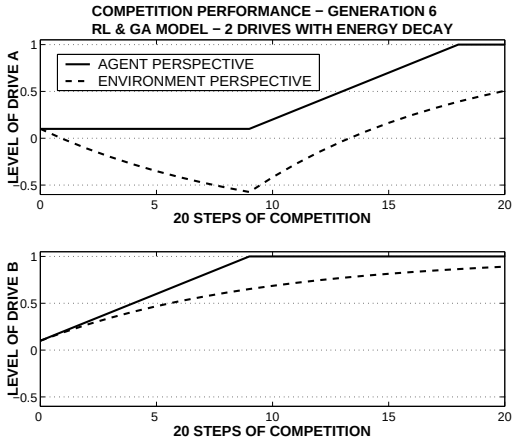
In Figure 4.5(b) the converged Q-Values of generation 6 are plotted. Different behaviours cooperate in order to push the levels of the drives towards their maximum level of 1.0 and the effect of the energy decay is to develop greedy agents that



(a) Average & Maximum Fitness



(b) Converged Q-Values



(c) Competition Performance

Figure 4.5: RL & GA Model with 2 Drives and Energy Decay

“anticipate” a possible decay on their drives and hence pursue behaviours that have a positive effect even when the target has been reached.

A typical competition performance together with its “realistic” counterpart that includes the energy decay, can be seen from Figure 4.5(c). Both drives manage to reach the goal state of 1.0 but the performance on Drive A would not be scored as high if the energy decay was also included in the competition.

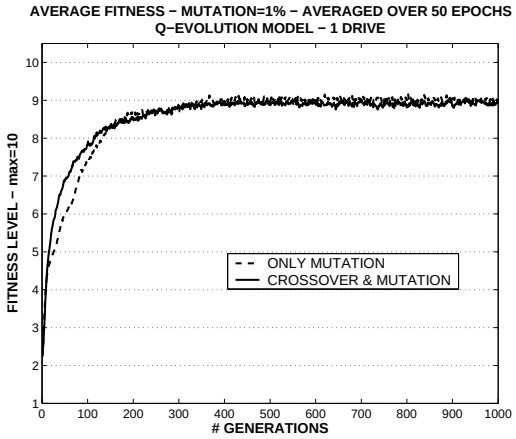
4.2 Second Model - Q-Evolution

This section reports the results obtained from the second model that was proposed. This model is directly evolving Q-Values and explores the capabilities and restraints of directly evolving the sense of valency without learning being applied in the lifetime of the agents. The standard four cases, of one or two drives and with or without energy decay, are considered and the results in most cases are averaged over 50 epochs. Two different GA settings are tested again, the first performing only mutation and the other one performing the standard GA setting with both crossover and mutation. Various mutation rates are considered.

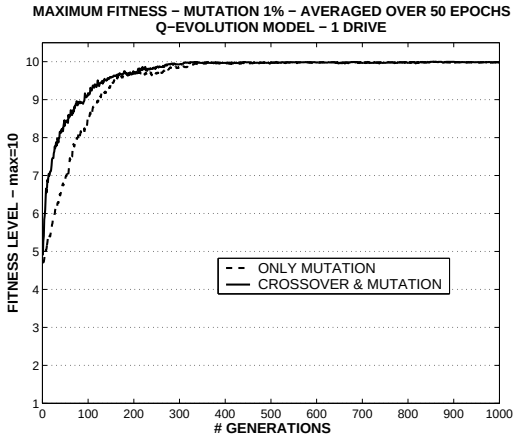
4.2.1 One Drive

The first case of the Q-Evolution model with single drive agents is presented in this section. Figures 4.6(a) and 4.6(b) show the average and maximum fitness development over generations for both settings with a mutation rate of 1%. The same metric is plotted in Figures 4.6(c) and 4.6(d) for a mutation rate of 8%. When the rate is high, the setting that has only mutation performs better but overall the highest average and maximum fitness is performed using both crossover and mutation with a mutation rate of 1%.

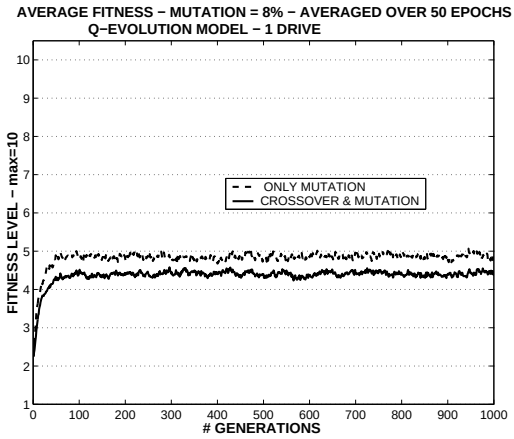
From Figures 4.6(e) and 4.6(f) typical *evolved* Q-values can be seen for all the four different settings. All of the examples are “perfect” solutions that achieve the maximum score of 10 in the competition. The difference between Q-Values for one action and another are not important and do not represent a relative valency since only the largest one is used and it has no importance by which margin an action is preferred



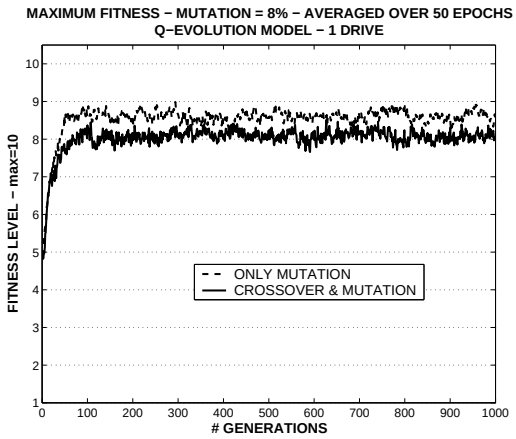
(a) Average Fitness - Mut 1%



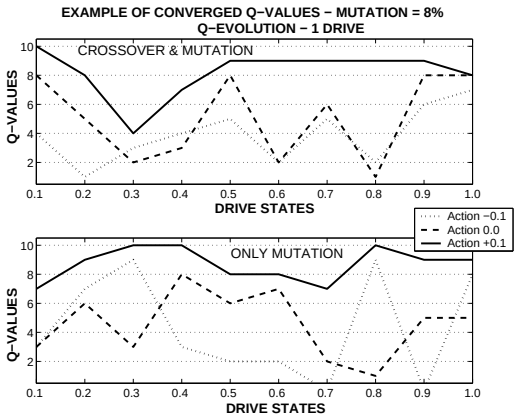
(b) Maximum Fitness - Mut 1%



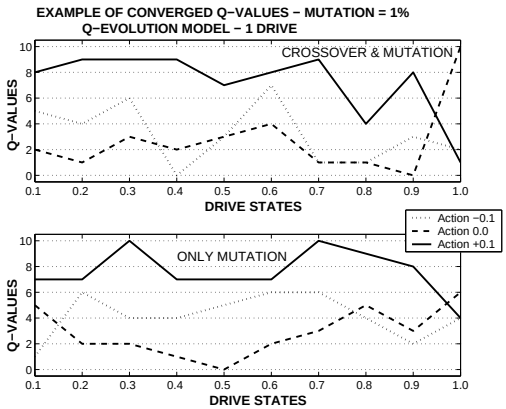
(c) Average Fitness - Mut 8%



(d) Maximum Fitness - Mut 8%



(e) Evolved Q-Values



(f) Evolved Q-Values

Figure 4.6: Q-Evolution Model with 1 Drive

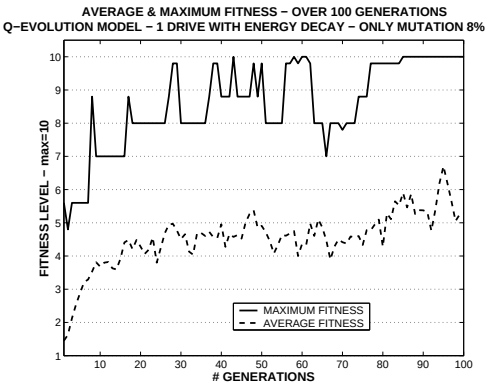
over the other. Also, there is no learning present in this model and hence no way that other actions can be selected (no exploration present).

4.2.2 One Drive with Energy Decay

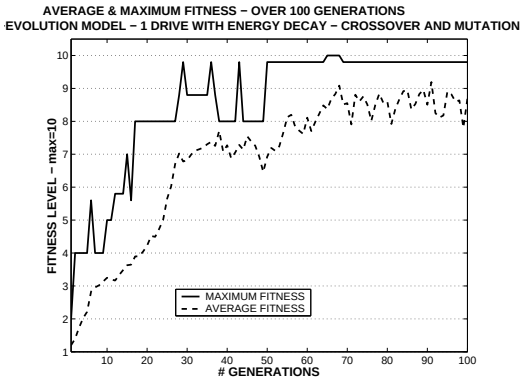
In this section we consider the inclusion of the energy decay function in the Q-Evolution model with 1 drive agents. This model is directly evolving the sense of valency (Q-Values) without any learning mechanism being used and hence the energy decay has to be applied on the competition. The competition steps were increased to 100 to allow both the agent to reach the goal state of its drives and also the actual energy level from the environment's perspective (since the agent perceives only discretized states) to reach the 1.0 level.

In Figures 4.7(a) and 4.7(b) the average and maximum fitness is presented for two GA settings. The first setting is using only mutation with a rate of 8% and the other is using both crossover and mutation with a mutation rate of 1%. As can be seen, the model using both crossover and mutation has a steeper increase of the average fitness and it reaches faster to a "perfect" solution.

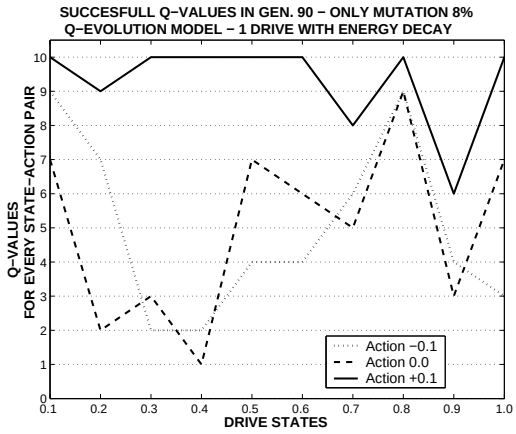
An example of successful Q-Values can be seen in Figure 4.7(c) and the corresponding competition performance in 4.7(d). The evolved Q-Values for action +0.1 are greater for every state and hence that action will be selected always. The drive level increase is discretized and sudden from the agent's perspective while from the environment's perspective is exponential and smooth.



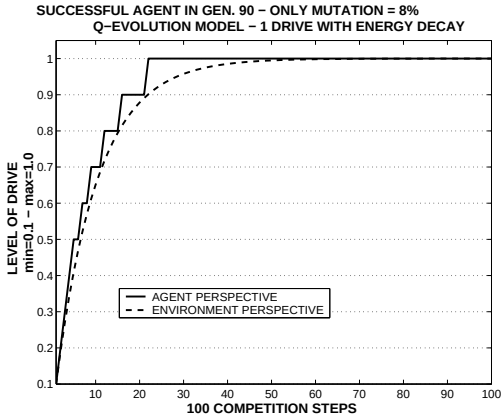
(a) Average & Maximum Fitness



(b) Average & Maximum Fitness



(c) Evolved Q-Values



(d) Competition Performance

Figure 4.7: Q-Evolution Model with 1 Drive and Energy Decay

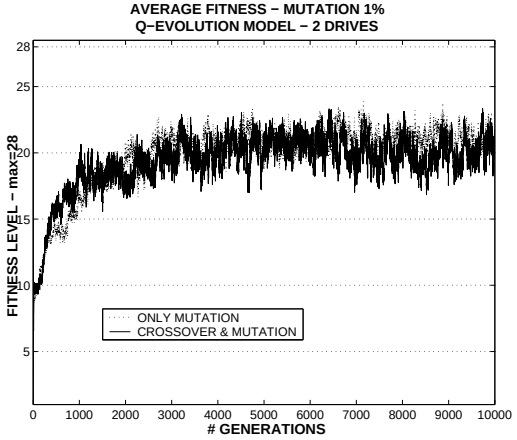
4.2.3 Two Drives

In this section we report results using the Q-Evolution model and agents with two drives. As the problem difficulty increases the model requires more generations in order to reach a satisfactory level of fitness. This becomes apparent in Figures 4.8(a) and 4.8(b), where the average and maximum fitness is presented for our two different settings, as the first “perfect” solution is achieved only after 2000 generations.

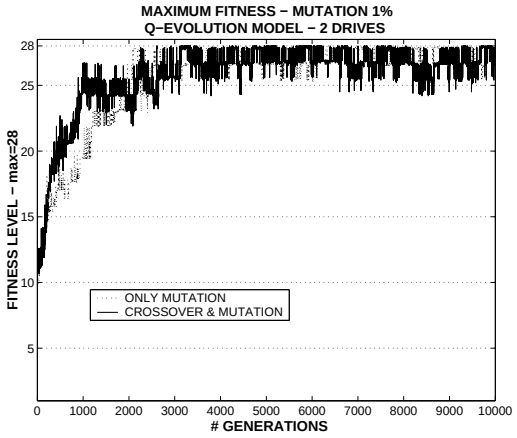
Two different fitness functions and competitions were tested with this model. The first one had a “double” competition in which the agent was using its Q-values from drive A for the first 10 steps and then its level was re-initialised and another 10 steps performed this time using the Q-Values from drive B. The level it reached on both occasions was used to calculate its fitness (c.f. Table 3.5).

Using that fitness function we guided evolution towards Q-Values, see Figure 4.8(c), that are dominant for the action that has the effect of +0.1 on both drives. That action was “awarded” the highest valency from both drives since they were both tested and not competing with each other. In Figure 4.8(d) the competition results are presented for a successful agent.

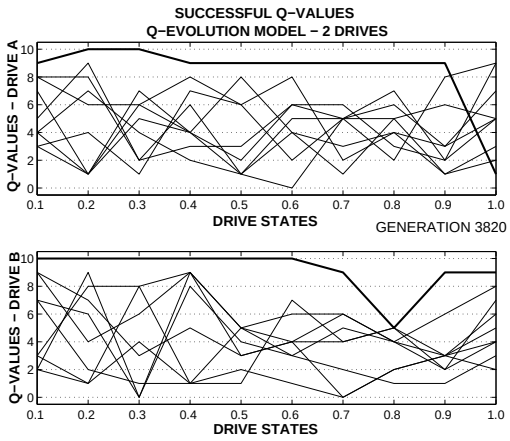
The other fitness function used was to consider a 10 step competition with both sets of Q-Values active. As it can be seen in Figures 4.8(e) and 4.8(f) the result was that the greater Q-Values always corresponded to the action with +0.1 effect on both drives. But this was not due to one drive taking over the other but the two drives cooperated in order to always choose the action with the maximum benefit for both. When the valency of that action was low in one of the drives then the other drive had the greatest valency on that same action overall.



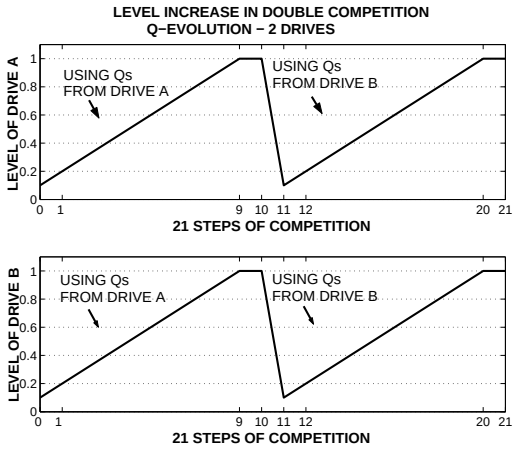
(a) Average Fitness



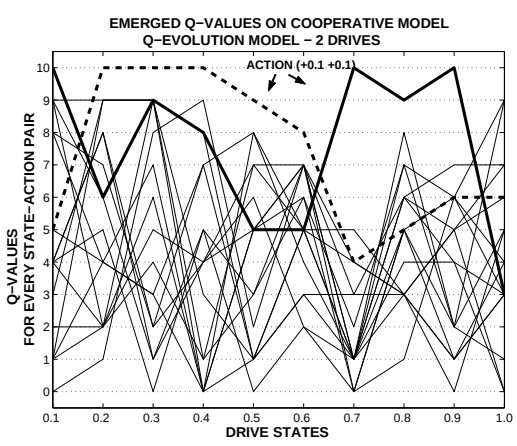
(b) Maximum Fitness



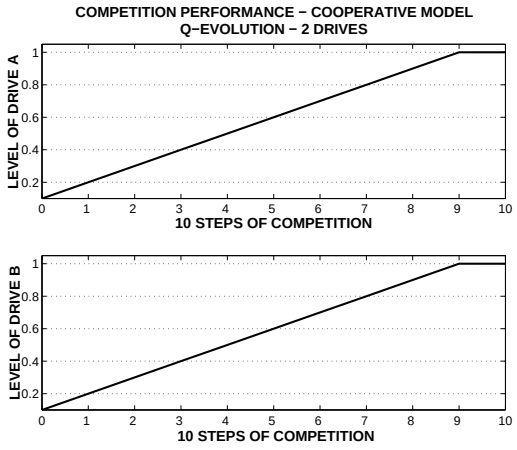
(c) Evolved Q-Values



(d) Competition Performance



(e) Evolved Q-Values



(f) Competition Performance

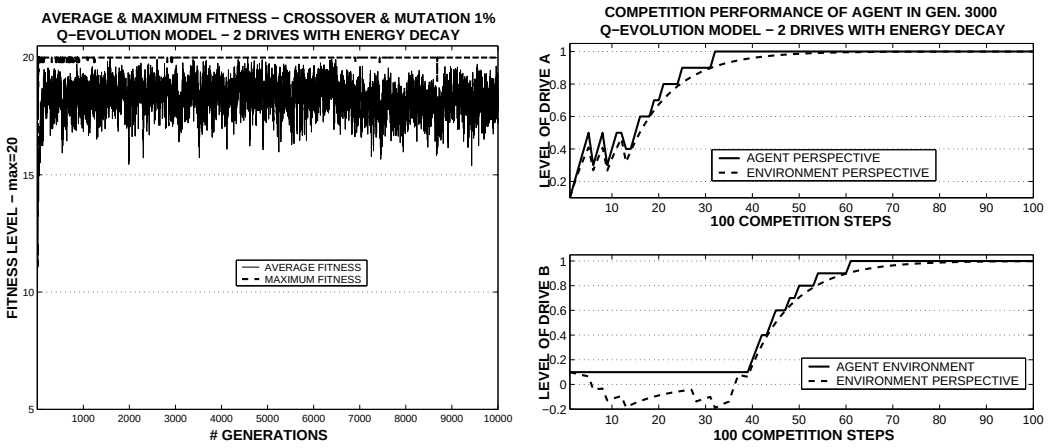
Figure 4.8: Q-Evolution Model with 2 Drives

4.2.4 Two Drives with Energy Decay

The results for the energy decay addition on the Q-Evolution model with two-drives agent are reported in this section. The fitness function determines the behaviour of our models. By using two different functions we report emerging behaviours and limitations of the model. The first fitness function tested was based only on the level of the drives at the end of the 100 steps of competition. The results can be seen from Figures 4.9(a) where the maximum and average fitness is plotted and 4.9(b) that presents a typical competition performance. The agent reaches the goal state by a combination of behaviours without "feeling the pressure" to select the best possible action for both drives and give a high valency on that.

Conversely, using a fitness function that depends on the level of the drives at the 22^{nd} step we apply a pressure for the agents to select that best possible action. This is because if the agents choose any other behaviour, apart the one that has a +0.1 effect on *both* drives, they will be unable to reach the goal state of 1.0 within these 22 steps (lower limit of steps needed when energy decay is acting) and hence receive a lower fitness. Thus with this fitness function we discriminate between possible solutions in reaching the goal state and we assign the highest fitness to the one that takes the drive levels of the agent to the goal state in the *quickest way possible*.

However, even after 50,000 generations the Q-Evolution model did not produce any perfect solution. That limitation of the model is attributed to the specific fitness landscape which is unable to lead evolution without learning towards that specific solution.



(a) Average & Maximum Fitness (b) Competition Performance

Figure 4.9: Q-Evolution Model - 2 Drives with Energy Decay

4.3 Third Model - RL and GA with NN

The results from the final model are presented in this section. This model is based on the RL and GA model that simulates both evolution and learning. Instead though of evolving the reward function, it evolves synaptic weights of a neural network that provides the reward function. The results report the fitness development along generations, typical synaptic weights evolved, reward functions produced from the network and the Q-values produced from the learning cycle.

4.3.1 One Drive

The single drive case was used in order to compare different neural network architectures and to choose the best one for our purposes. The difference between the architectures tested was the number of hidden units the network should have. Figures 4.10(a) and 4.10(b) show the average and maximum fitness for four different architectures and over 50 generations of evolution, is presented. The best performance is observed from the network with the two hidden units and next in performance is the network with six

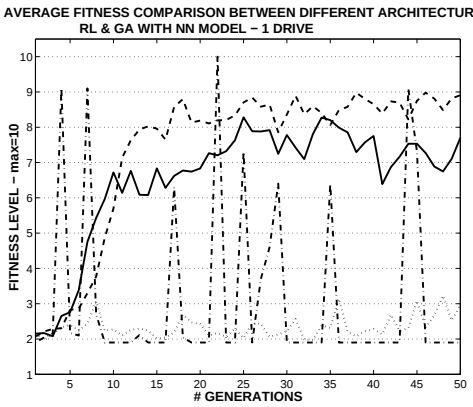
hidden units. The network without hidden units cannot model an appropriate reward function continuously and the network with six hidden units but with a mutation rate of 8% is unstable.

On the basis of a "rule of thumb", the number of hidden units should be more than the inputs but less than twice the number of inputs, the architecture with six hidden units and a mutation rate of 1% was chosen. The converged Q-Values, based on the reward from the network, are plotted in Figure 4.10(c). The action with the +0.1 effect has the highest valency in all states except the last one where it is equal with the 0.0 action since they have the same effect on the drives.

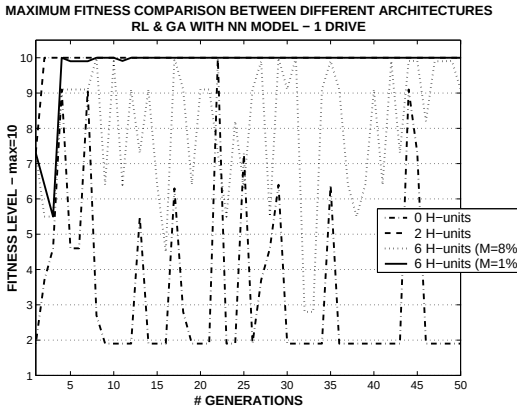
Typical values for the synaptic weights are presented in Figure 4.10(d) and the successful competition performance can be seen from Figure 4.10(e). A typical reward function produced from the neural network is reported in Table 4.2. The network came up with a familiar reward function that was mentioned in Chapter 2 in the labyrinth example. All state transitions are negatively reinforced with -1 except the final (looped) transition from the goal state to itself which receives a positive reward of +1.

	Ending State s_{t+1}										
Starting State s_t	*State*	*0.1*	*0.2*	*0.3*	*0.4*	*0.5*	*0.6*	*0.7*	*0.8*	*0.9*	*1.0*
	0.1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	0.2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	0.3	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	0.4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	0.5	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	0.6	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	0.7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	0.8	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	0.9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	1.0	-1	-1	-1	-1	-1	-1	-1	-1	-1	**1**

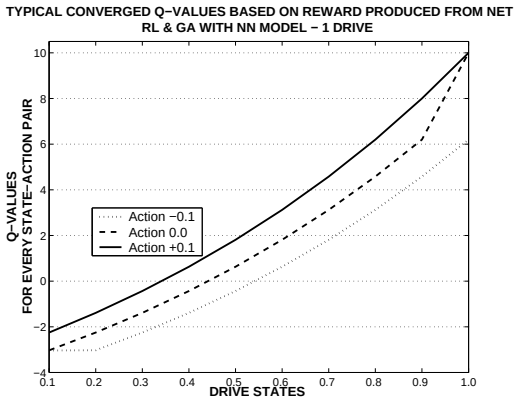
Table 4.2: Example of a successful reward function



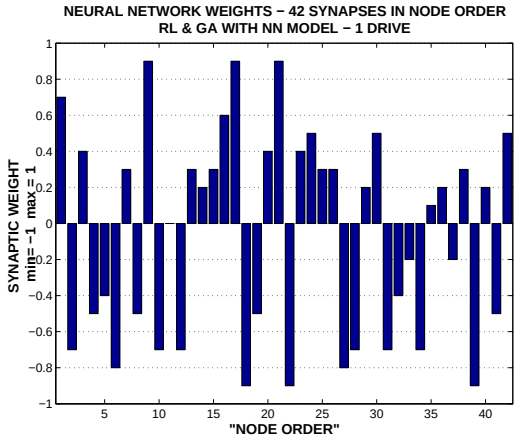
(a) Average Fitness



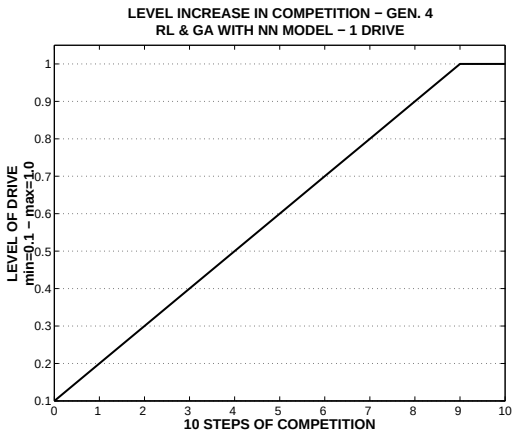
(b) Maximum Fitness



(c) Converged Q-Values



(d) Evolved Synaptic Weights



(e) Best Performance

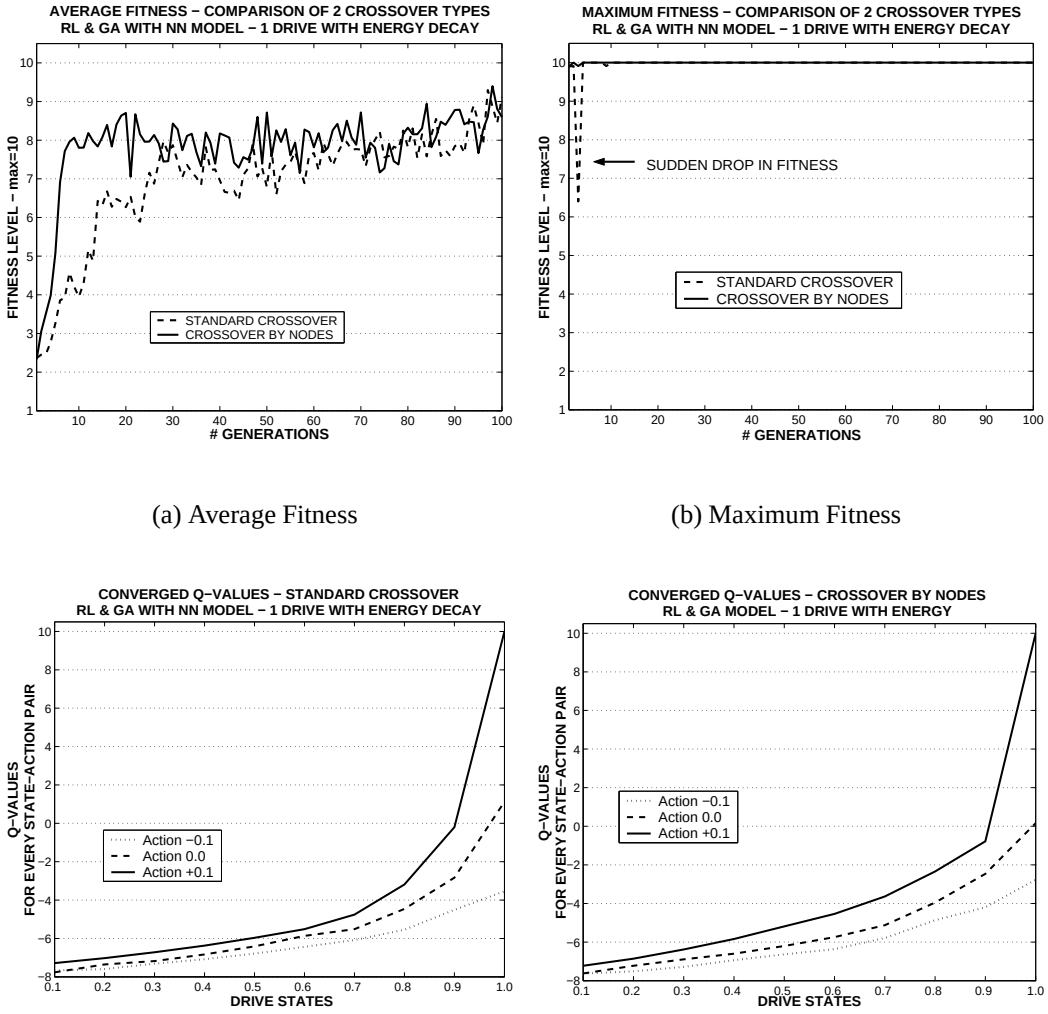
Figure 4.10: RL & GA Model with NN - 1 Drive

4.3.2 One Drive with Energy Decay

This case considers the addition of energy decay on the model with single drive agents. The decay function is applied, similarly to the first model, only on the reinforcement learning cycle and not on the competition. In Figures 4.11(a) and 4.11(b) the average and maximum fitness for two different types of crossover is considered. The standard crossover refers to the typical single-point crossover that was used so far in our models whereas the “crossover by nodes” refers to crossover that is performed between groups of weights that relate to a specific node. The results indicate that crossing over groups of weights between nodes produces better performance on the average fitness. The increase is steeper and within only 8 generations it reaches a fitness level of 8/10.

From Figure 4.11(b) we can observe a typical sudden drop on the maximum fitness around the third generation. That drop could imply that a building block (e.g the last row of Table 4.2) of the solution has not spread through the population yet and the behaviour of the population is still unstable. The sudden increase of the average fitness after the third generation supports the claim that only after the sudden drop the building block starts to spread into the genes.

The converged Q-Values for both crossover settings are plotted in Figures 4.11(c) and 4.11(d). The expected effect of the energy decay is present with the Q-Values for action +0.1 being completely separated and higher than the rest of the actions even at the last state.



(a) Average Fitness

(b) Maximum Fitness

(c) Converged Q-Values

(d) Converged Q-Values

Figure 4.11: RL & GA Model with NN - 1 Drive with Energy Decay

4.3.3 Two Drives

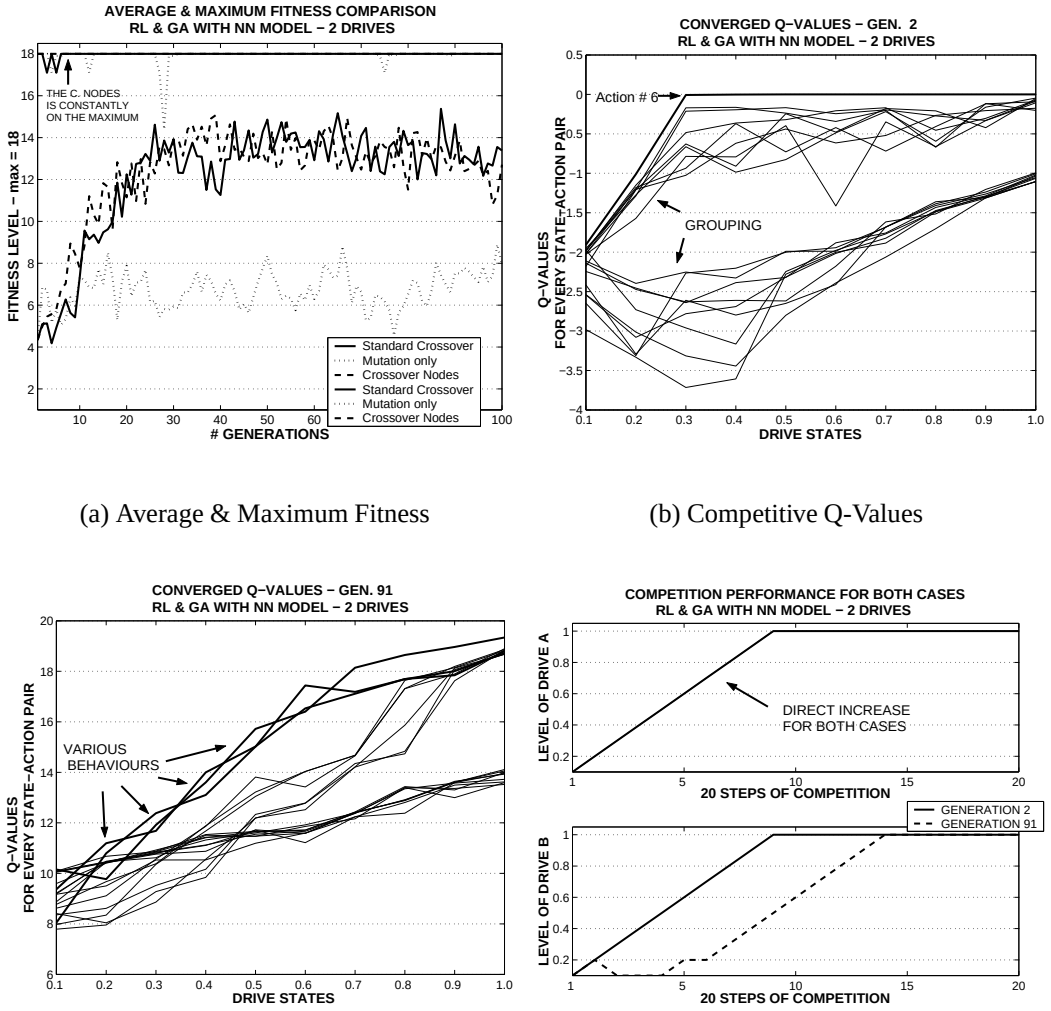
This section reports results for the third model in the case of agents with two drives. The addition of the second drive on the model requires the extra addition of an input node to the neural network in order to produce the reward function for both drives.

In Figure 4.12(a) a comparison is made between three different GA settings on

the average and maximum fitness. The two crossover types are compared against a case where only mutation is used in order to evolve the synaptic weights. As it can be seen, the crossover and mutation cases perform better with higher averages and stable performance on the maximum fitness. The setting that uses only mutation achieves a very low average due to the random nature of mutation and although it manages to reach perfect solutions from the beginning, in certain cases the maximum fitness drops suddenly.

Two very different solutions are produced with the converged Q-Values in Figures 4.12(b) and 4.12(c). The first case from the second generation develops a grouping of Q-Values for different drives. One drive has the highest Q-Values for all actions and hence it becomes the "dominant" drive with its selected behaviour becoming the behaviour executed. The selected behaviour of that drive is always to perform the action that has +0.1 effect (6^{th} Action) on both drives and hence both drives reach the goal state with the minimum number of steps.

The second solution observed is the cooperative one, where a multitude of behaviours are executed towards reaching the goal state, see Figure 4.12(c). In that case three different behaviours interchange by having different valency in different states. The competition performance for both cases is displayed in Figure 4.12(d). Note how the competitive behaviour reaches the target faster and more efficiently than the cooperative behaviour.



(a) Average & Maximum Fitness

(b) Competitive Q-Values

(c) Cooperative Q-Values

(d) Competition Performance

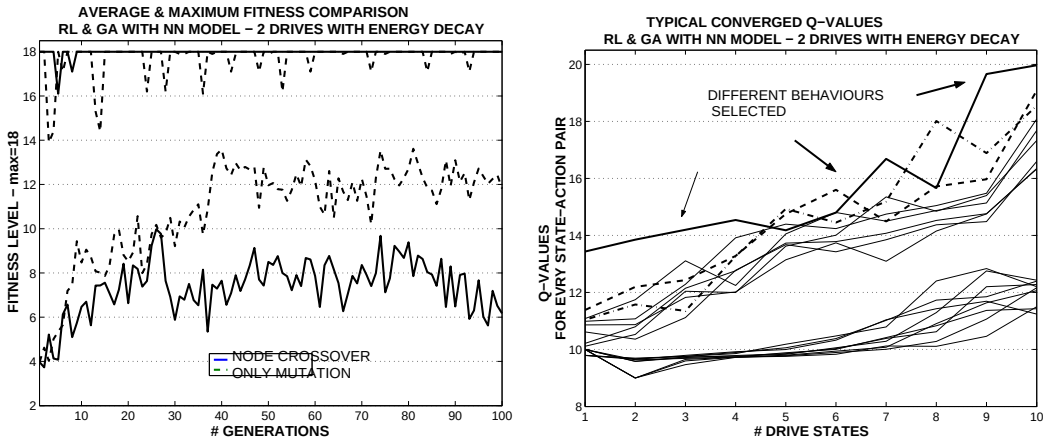
Figure 4.12: RL & GA Model with NN - 2 Drives

4.3.4 Two Drives with Energy Decay

The final case case considered for the third model is the inclusion of the energy decay function on the two-drive agents case. In Figure 4.13(a) the average and maximum fitness development is compared between the node crossover setting and the case with only mutation. Although the node crossover setting is still more consistent in sus-

taining a perfect solution in the population, its average fitness levels are much lower than the only mutation case. The increase in performance levels for the case with only mutation can be accredited to the energy decay introduction that applies some stochasticity into the problem and makes the reward function easier to be approximated by the network.

Figure 4.13(b) shows typical Q-Values developed on the basis of the reward function that was produced by the neural network. The solutions produced with the standard fitness function, are usually of the cooperative style with many behaviours cooperating towards the goal and only rarely a single behaviour takes over. However, as it was demonstrated before, by applying selection pressure through different fitness functions we can exert a more competitive behaviour.



(a) Average & Maximum Fitness

(b) Converged Q-Values

Figure 4.13: RL & GA Model with NN - 2 Drives with Energy Decay

4.4 Unstable Environment - All Models

In this section we report results for the behaviour of the three models on an unstable environment. The rate of instability was examined on four different values (i.e switching the effect of an action in the drives every 25, 13, 7, 1 generations) in agreement with

the setting used by [Mclean, 2001]. In most cases only the extreme case of a change occurring every generation is reported because if our models can handle that setting they will be able to handle less dynamic environments. The relationship between rate of environmental change and performance is explored by comparing performances for all different rates in a single case. The unstable environment is considered for all four cases (one or two drives and with or without energy) in order to demonstrate that the observations made are independent of the problem's difficulty level.

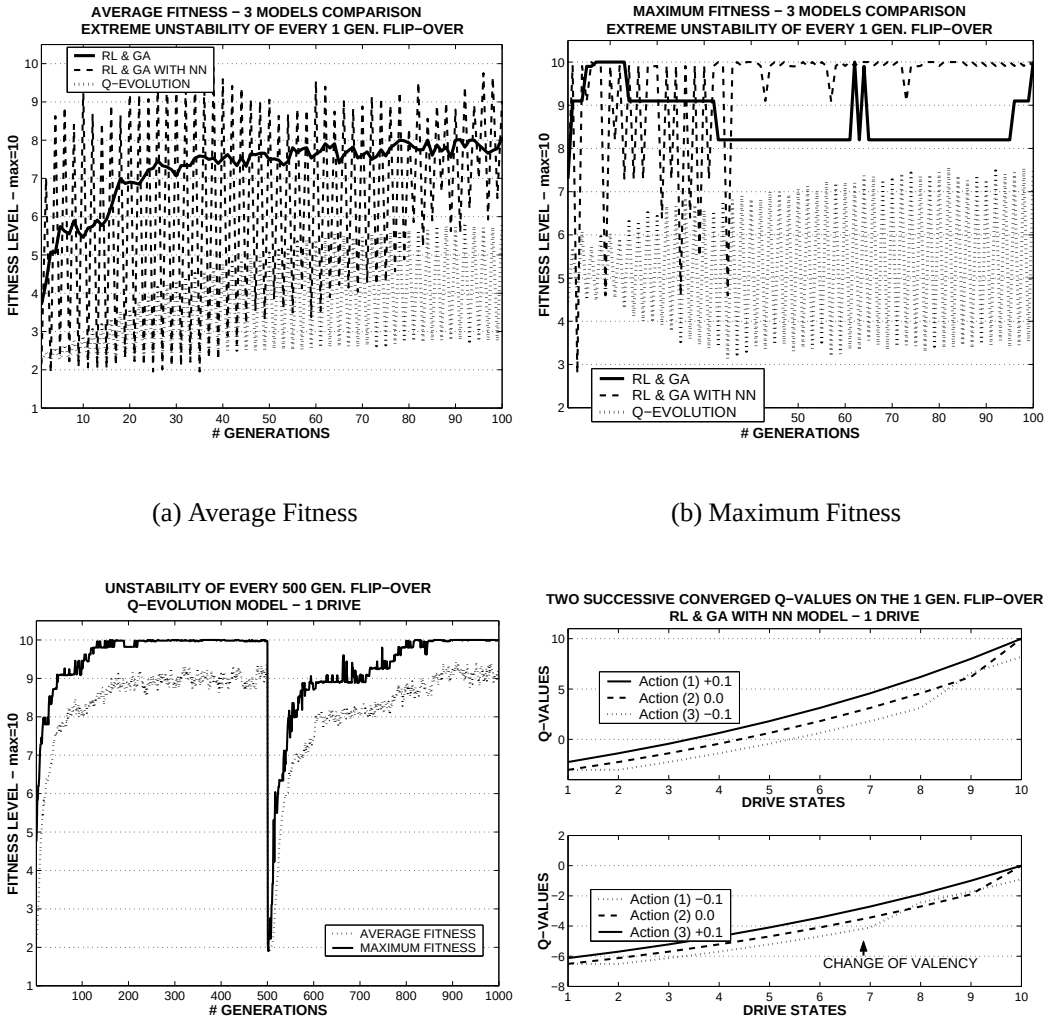
The results presented concentrate on the maximum and average fitness development and on typical Q-Values (valencies) for successive generations in order to demonstrate how the inversion of action effects impacts on the development of a successful sense of valency.

4.4.1 One Drive

The first case considered is the single drive models. In Figures 4.14(a) and 4.14(b) our three models are compared on the extreme unstable environment where in every generation the effect of actions inverts. The first model (RL and GA) that incorporates both learning and evolutionary mechanisms is maintaining a constant increase on its average fitness development and manages to reach perfect solutions on the maximum fitness, even successively.

The second model (Q-Evolution) is using only an evolutionary mechanism that proves unable to keep up with such an extremely unstable environment. Both the average and maximum fitness development is fluctuating with an increasing variance until a point where the variance settles to an extreme. the model cannot cope with the every generation change and fails to provide a proper set of Q-Values that can lead the agent to the complete satiation of its drives.

The third model (RL and GA with NN) considered, is also fluctuating between two different average fitness values , Figure 4.14(a), but the variance is decreasing and the mean of the fluctuation increases and approaches a value of 8/10. Also, the maximum fitness development is fluctuating initially for the first 40 generations between two values but then it settles and provides successive perfect solutions.



(a) Average Fitness

(b) Maximum Fitness

(c) Q-Evolution Model

(d) Converged Q-Values

Figure 4.14: Unstable Environment - 1 Drive

The failure of the second model and the success of the other two models can be explained on the basis of their architecture. The Q-Evolution model directly evolves the sense of valency and hence when the environment changes it has to re-start evolving that sense from the beginning as it can be seen in Figure 4.14(c) where even a 500 generation flip has the same effect. The other two models though are evolving the reward function with respect to the internal states of their drives which are unaffected

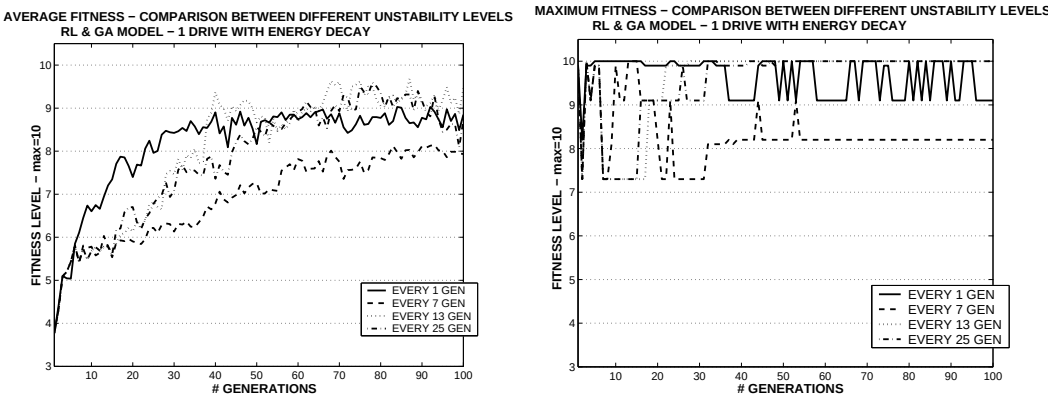
by any environmental change (the drives goal state is always to reach level 1.0). Hence although the effect of actions is changed, the learning mechanism can cope since it is based on the reward that it receives when satisfies its drives. So, it will learn to assign greater valency to the appropriate action, as it can be seen from Figure 4.14(d).

4.4.2 One Drive with Energy Decay

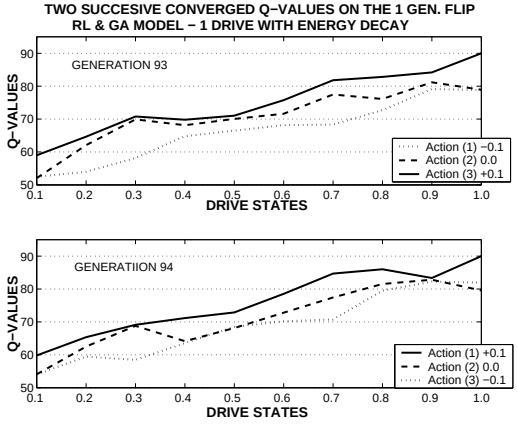
In this section we futher examine the unstable environment effect on the first model and with the energy decay applied on single drive agents. In Figures 4.15(a) and 4.15(b) the maximum and average fitness development is plotted for all four different rate of change settings. As we can see the model can cope with all the cases having a better performance on the cases where the change occurs every 25 and 13 generations. Although the average fitness is increased faster with the every one generation change, it settles to a lower level after 100 generations. Similarly it manages to reach a perfect solution within the first 5 generations but it does not continue to perform stable as it flips between different high fitness levels in contrast with the other two cases.

The case where the change occurs every 7 generations is performing worse than the rest although its average fitness is still increasing without extreme instabilities There is no explanation offered for that effect. To resolve that point we would need average performances over many runs and to test also all the flipping rates between 1 and 7, which was not feasible within our time framework.

Finally, in Figure 4.15(c) two successive and successful Q-Value results are presented. As it can be seen the same action has different effect each time and hence different Q-Values. The energy decay effect is to give greater valency on the action with effect +0.1 even in the last state and hence influence the agents towards greedy behaviour.



(a) Average Fitness (b) Maximum Fitness



(c) Successive Q-Values

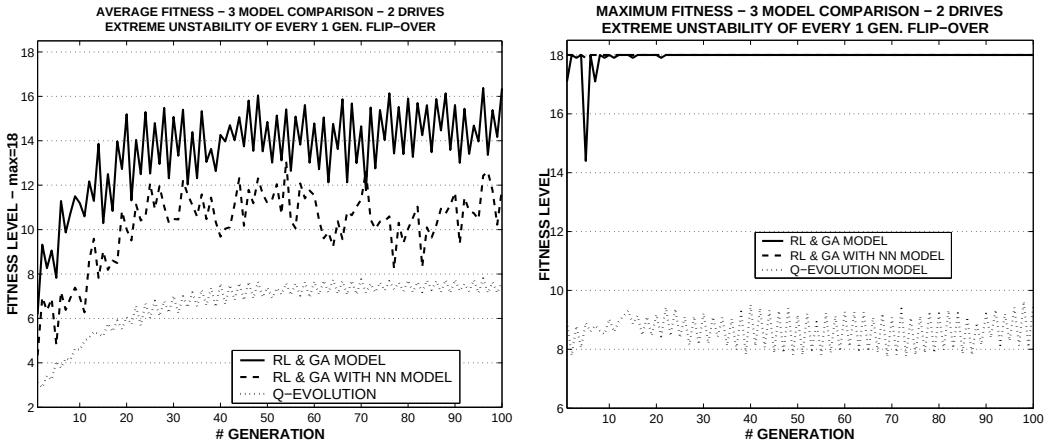
Figure 4.15: Unstable Environment - 1 Drive with Energy Decay

4.4.3 Two Drives

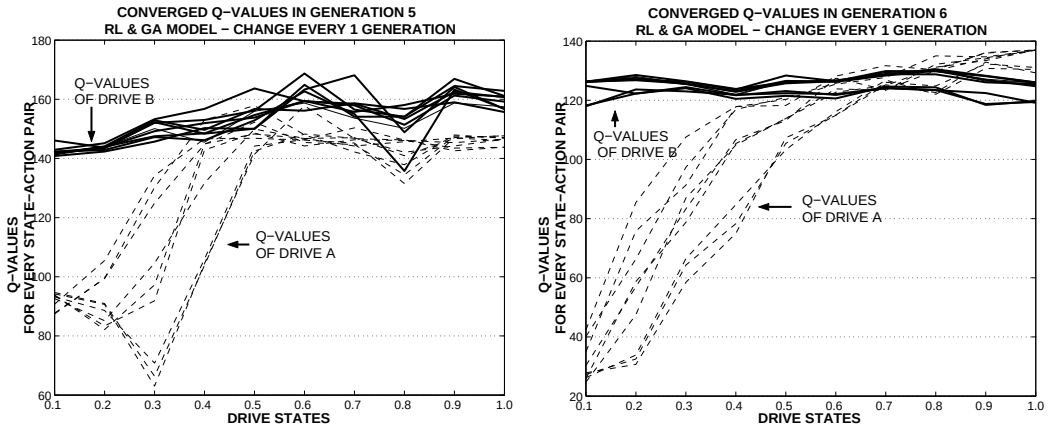
The addition of the second drive in our models within an unstable environment is reported. The performance of the three models is compared over the average and maximum fitness development in Figures 4.16(a) and 4.16(b). The first and third model that use both learning and evolution are successful with the “RL and GA” model achieving a higher average fitness but with the “RL and GA with NN” model avoiding fluctua-

tions at the initial stage. The Q-Evolution model is completely unstable and not able to recover.

Typical successive Q-Values are presented in Figures 4.16(c) and 4.16(d).



(a) Average Fitness (b) Maximum Fitness



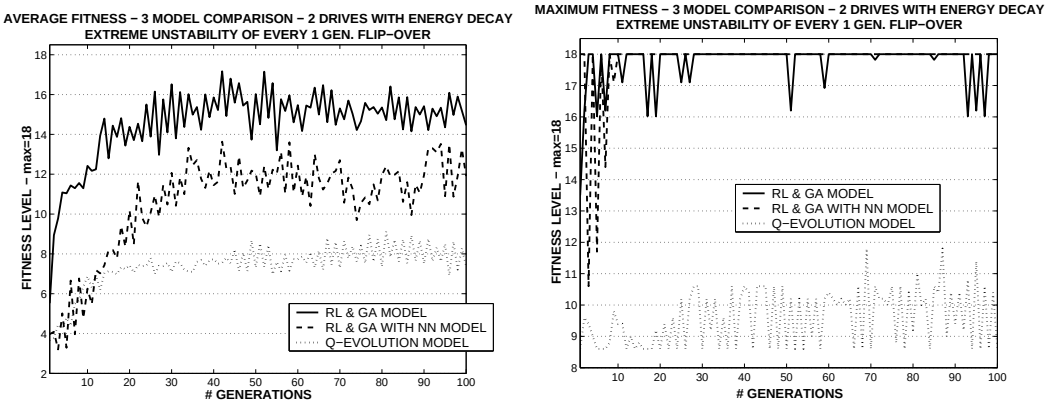
(c) Converged Q-Values (d) Converged Q-Values

Figure 4.16: Unstable Environment - 2 Drives

The behaviour selection is similar to the results from the stable environment with grouping of Q-Values depending on the drive they belong to and also with a multitude of behaviours being selected when the fitness function and the competition style are relaxed.

4.4.4 Two Drives with Energy Decay

Finally, all the models with two drive agents and energy decay applied are tested on the unstable environment. The average and maximum fitness development for the three models is presented in Figures 4.17(a) and 4.17(b). The same effect is observed with the Q-learning model being unable to cope with the instability of the environment and with the two models that incorporate both learning and evolutionary mechanisms outperforming it.



(a) Average Fitness (b) Maximum Fitness

Figure 4.17: Unstable Environment - 2 Drives with Energy Decay

Chapter 5

Discussion

5.1 Stable Environment

This study aims at providing some understanding of the mechanism responsible for the animal sense of valency and of its evolution throughout time. We presented three different models that are able to evolve successful valency associations and have tested them on different settings, cases and environments.

The *first model* simulated both learning and evolution by evolving the reward function and learning the valency of actions during the lifetime of agents based on that function. The results indicate that two parameters are important for the "shaping" of what makes a successful reward function: the *discount factor* and the *fitness function*.

The *discount factor* described the present value of future rewards and hence it defined how strict a reward function must be in order to be successful. For low values, it required a finely tuned reward function - hence making the task of evolution harder - that leads the agent in every state and for high values it only required a higher average reward value towards the goal state.

The *fitness function* was important for all three models since it applied pressure towards a desired behaviour. It guided evolution towards a specific goal and hence it was used to explore different behaviours and reward functions responsible for those behaviours. It defined the exact fitness of every reward function on the basis of the behaviours that were developed.

In general the fitness function defines the available knowledge that an agent can have over the environment. The reward functions evolved making use of that knowledge and hence depended on the fitness function. The use of a "relaxed" function produced rewards and valencies that exerted a multitude of behaviours to reach the target. When the fitness function became strict (incorporated more knowledge of the environment) then the necessary reward function and sense of valency was evolved and learnt in order to identify the *best* behaviour. The need for a more informative fitness function in order to assign the greatest valency to the single best behaviour arises from the fact that this association is unique (hence difficult to evolve) and constitutes a singularity within the problem domain.

The *second model* simulated only evolution by directly evolving the valency of behaviours. From the results on the stable environment it is observed that although the model can evolve successful valencies, the number of generations needed increases rapidly as the problem difficulty increases. The magnitude of the required generations (e.g 3,800) is beyond any biological plausibility[1] since on average there are 80 generations within 2,000 years. The model is computationally fast and efficient but not as informative on the relative valency between behaviours since there is no learning incorporated.

The *third model* was an advancement on the first model in order to deal with the dimensionality problem. The neural network provided the reward function while its synaptic weights were evolved through generations of agents. The reward functions produced from this model were again indicative of the nature of our learning algorithm. The negative reinforcement of all state transitions except the transition from the final state to itself, is the shaping of the reward function by the need of the "far-sighted" Q-Learning rule to seek high future rewards.

For all the models, the cases using crossover and mutation with a small mutation rate of 1% proved to perform better than just mutation and especially as the difficulty of the problem increased. This was because mutation is a random process performing local search around solutions and not retaining and passing over structured blocks of information. The results indicate that a high mutation rate destroys already perfect

[1]For humanoid evolution at least. As I was informed there are examples of organisms that could easily justify such evolutionary magnitudes - e.g. for bacteria there are 3 generations per minute.

solutions and causes instability on the maximum fitness performance. On the other hand, small mutation rate introduced the necessary variation among solutions and did not obstruct the building blocks of the crossover mechanism.

Finally, the introduction of the energy decay function caused greedy behaviours from the agents to be exerted and ensured that the agents kept on pursuing and associating greater valency to the positive-effect action, even in the goal state.

5.2 Unstable Environment

The results from the unstable environment make a strong distinction between our models. The two models that simulate both evolution and learning, can withstand even the most extreme instability in the effects of actions. They are still able to evolve correct valencies every generation and hence pursue the necessary behaviours to reach the goal state. However, the Q-Evolution model, that simulates only evolution, is completely unsuccessful even in low instability environments.

Conversely to the results reported by [Mclean, 2001], where by flipping food to poison[2] and vice versa an unrecoverable instability in average and maximum fitness is produced, our two models utilising both learning and evolution can retain their performance . This is because both models evolve the reward function which is not affected by any such instability. The reward function dictates the reinforcement the agent receives for state transitions on its drives that remain unaffected from such instabilities on action effects.

For example, imagine one drive that could account for level of poison eaten by the agent. The selection pressure (fitness function) would evolve a reward function that would reward state transitions towards the minimum level (since its fitness would be the highest). Thus, during the lifetime of the agent, any action leading to increasing the "poison" drive level would have a low valency and any action that reduced the level would have a high valency (opposite to our standard convention so far that the 1.0 level is the goal state). Our two models do not have any pre-association towards spe-

[2] In one generation certain "objects" are food and others are poison with the agent learning to discriminate between the two. In the next generation (extreme case) the food objects become poison and opposite with the effect of agents eating the wrong thing and hence having suddenly low fitness.

cific actions (eating food is good) they only have a pre-association (from the previous generations) towards what effect is “good” or “bad” for them (increasing the nutrition level is good). That is why the agents will not go about trying to increase their nutrition drives by eating poison.

That difference is also the reason for the failure of the Q-Evolution model in the unstable environment. The model is directly evolving Q-Values and hence accredits a high valency on specific state-action pairs. So when it is applied on the unstable environment it has to unlearn and re-learn the correct valencies.

Finally, the Baldwin Effect is not explicitly present in this problem domain for the two models that incorporate learning mechanisms. By using the evolved reward function, or the reward produced from the evolved synaptic weights, our models construct valencies through the learning mechanism. These valencies are used to decide on the fitness score of every agent. Therefore, reward functions that can lead to correct valencies will be favoured. However, there is no explicit cost of learning incorporated in our models and hence no pressure for specific modifications to occur in the genotype in order to help the learning procedure and reduce the cost of learning.

It is worth noticing that a similar effect can be caused by the low discount factor case. The agent becomes myopic hence it will evolve reward functions that are descriptive for *every* state and therefore *predefining* valency for state transitions (instead of a reward function that needs learning to derive valencies). In that way we can claim that the genotype evolves to incorporate knowledge that determines the learning of the phenotype.

Chapter 6

Conclusions and Future Work

In this final chapter we present the main conclusions derived from this study and we propose future interesting directions for further research on the topic.

6.1 Conclusions

The general conclusions are that it is possible to evolve a sense of valency and that it is necessary to do so in order for an autonomous agent to be really autonomous and able to withstand changes in its environment. We also conclude that Learning is vital for such a valency mechanism to successfully cope with instabilities. Learning adjusts valency within lifetime by "sampling" the effect of actions from the environment.

Furthermore, Learning overcomes the problem posed from unstable environments due to the separation of genotype and phenotype. The genotype of the agents represents possible solutions (reward functions) for *basic needs* (the goal state), while the phenotype represents valencies (Q-values) for *paths toward the basic need*.

Finally, we conclude that Darwinian Evolution alone cannot cope with such instabilities since it addresses a larger timescale than the lifetime of agents. The main conclusions drawn are presented according to the specific area of the valency problem.

Technical Conclusions

- **Genetic Operators** - Mutation alone proved to perform worst in high-difficulty environments giving low average fitness for the population due to its random nature. Crossover together with low rate mutation gave the best performances both on the individual and the population level because this method creates building blocks of information and also allows for random non-destructive variation.

- **Fitness Landscape** - When the design of the fitness function was very “strict”, learning was found (In the double drive experiments) to be necessary in order to smooth the fitness landscape and provide evolution with an “easier path” towards the solution.

- **Dimensionality** - The dimensionality problem of increasing the number of drives was tackled with the use of a neural network approximating the necessary reward function.

Modeling Conclusions

- **Fitness Function** - The nature of the fitness function defined the reward function and the valency developed from the models. It also defined the relative differences in the level of valency between the available actions.

- **Discount Factor/Rate** - The magnitude of the discount factor directly influences the reward function. Low values require a more refined reward function and hence more generations whereas high values need a reward function that is providing a higher expected future reward at the goal state.

- **Effect of Learning** - For unstable environments it is necessary to incorporate a learning mechanism into the model in order to sample the changes that have occurred and react accordingly.

- **Evolution** - Evolution alone cannot cope with unstable environments when it is directly evolving the valency. It has to unlearn and relearn the new environment and thus needs large generation intervals between changes.

Conclusions on Valency

- **Common Scale** - Valency has a common scale in order to compare different effects of actions and rate them accordingly. It also depends on the state of the drive.
- **Separation** - The separation of the genotype (reward function/synaptic weights) from the phenotype (Q-Values) proved to be the solution in order to cope with instabilities of the environment. The learning mechanism is responsible for the phenotype and is therefore able to modify it according to the environment feedback. The evolution is responsible for the genotype and only needs to encode basic information, e.g. what is the goal state on the drives.
- **Energy Decay** - The energy decay proved necessary for further separation of valencies that under previous conditions were considered equal due to actions having the same apparent effect on the drives.

6.2 Future Work

This study has offered an overview on the consept of valency. A major point is whether the animal's sense of valency derived from evolution at the first place. Is the basic element that gives rise to a correct sense of valency coded in our genes? Is the satiation of drives rewarded from a hard-wired element or is it a genome that has evolved through us and pre-defines the goal state in our "drives"? Those questions and many more give rise to specific work towards a more complete theory. Future proposals are:

- To increase the number of drives and the action repertoire while using a neural network for the dimensionality problem in order to explore further behaviours.
- Instead of learning Q-Values (valencies) for every drive and competing between drives, combine the valencies of drives for an action into an overall valency.
- Explore the effect on the reward function when different learning rules are used, especially the ones considered biologically plausible.

- Another case would be to evolve reward functions for complex environments and then pass that reward function to a robot in such an environment.
- Instead of using a separate competition and hard bounds on the drives score the agents during their lifetime and "kill" any agents surpassing a certain limit away from the homeostasis of the drives. This will be a further step toward a more biologically plausible model.
- Introduce dependency of certain drives on the levels on other drives and stochastic effects of actions.
- Instead of predefining the drives an agent can have, allow it to evolve them.
- Study the role of dopamine on animal behaviour and introduce a model that simulates such a valency mechanism. That would lead to a better understanding of the valency mechanism and perhaps identify how the animal "reward function" operates.
- Include cost of learning into the models in order to observe a strong Baldwin Effect and analyse its effect on valency and the reward function.
- Finally, from a designer's point of view we have moved the problem of pre-designing the reward function to the need of pre-designing the fitness function. However, it is not a vicious circle; the reward function deals with the experience within lifetime (as part of Learning) and the fitness function with the advancement in the generation level. An interesting proposal is to research ways that an agent can approximate the fitness function from the environment.

Food for Thought

The most important aspect that defines the evolved reward function and thus the sense of valency is the fitness function. Through "competition" knowledge,requirements about the environment are passed on to the agent. However since we humans have created environments within the natural environment, we are defining a fitness function on our own. At a social level it is hard to define who is "fit" and who is not since our

ability to reproduce does not depend on that. In the contrary, highly educated or rich people (that are enjoying a status of being fit) usually pass on their genes less likely than poor and uneducated people that create big families. What is our “fitness function” and its consequence and is our sense of valency changing?

Bibliography

[Ackley and Littman, 1991] Ackley, D. and Littman, M. (1991). Interactions between learning and evolution. In Langton, C., Taylor, C., Farmer, D., and Rasmussen, S., editors, *Proceedings of the Second Conference on Artificial Life*. California: Addison-Wesley.

[Baldwin, 1896] Baldwin, J. M. (1896). A new factor in evolution. *The American Naturalist*, 30 (June 1896):441–451, 536–553.

[Batali and Grundy, 1996] Batali, J. and Grundy, W. N. (1996). Modelling the evolution of motivation. *Evolutionary Computation*, 4(3):235–270.

[Darwin, 1859] Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection or The Preservation of Favoured Races in the Struggle for Life*. John Murray publishers of London.

[Daw, 2003] Daw, N. D. (2003). *Reinforcement learning models of the dopamine system and their behavioral implications*. PhD thesis, School of Computer Science, Carnegie Mellon University.

[Harvey, 1996] Harvey, I. (1996). Is there another new factor in evolution? *Evolutionary Computation*, 4(3):313–329.

[Hinton and Nowlan, 1987] Hinton, G. E. and Nowlan, S. J. (1987). How learning can guide evolution. *Complex Systems*, 1:495–502.

[Lamarck, 1809] Lamarck, J. B. D. (1809). Philosophie zoologique.

[Mayley, 1996] Mayley, G. (1996). Landscapes, learning costs, and genetic assimilation. *Evolutionary Computation*, 4(3):213–234.

[Mclean, 2001] Mclean, C. B. (2001). Design, evaluation and comparison of evolution and reinforcement learning models. Master's thesis, Department of Computer Science, Rhodes University.

[Mendel, 1866] Mendel, G. J. (1866). Experiments on plant hybridization. *Proceedings of the Natural History Society*.

[Miglino et al., 1995] Miglino, O., Lund, H. H., and Nolfi, S. (1995). Evolving mobile robots in simulated and real environments. *Artificial Life*, (2) 4:417–434.

[Mitchell, 1998] Mitchell, M. (1998). *An Introduction To Genetic Algorithms*. A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England.

[Nolfi, 1994] Nolfi, S. (1994). Learning and evolution in neural networks. *Adaptive Behavior*, (3) 1:5–28.

[Nolfi, 1999] Nolfi, S. (1999). How learning and evolution interact: The case of a learning task which differs from the evolutionary task. *Adaptive Behavior*, (7) 2:231–236.

[Schultz et al., 1997] Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275:1593–1599.

[Shizgal, 1997] Shizgal, P. (1997). Neural basis of utility estimation. *Current Opinion in Neurobiology*, 7:198–208.

[Spier and McFarland, 1998] Spier, E. and McFarland, D. (1998). Learning to do without cognition. In Pfeiffer, R., Blumberg, B., Meyer, J.-A., and Wilson, S., editors, *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behaviour, SAB98*, pages 38–47. MIT Press.

[Sutton, 1988] Sutton, R. S. (1988). Learning to predict by the method of temporal difference. *Machine Learning*, 3(1):9–44.

[Sutton and Barto, 2002] Sutton, R. S. and Barto, A. G. (2002). *Reinforcement Learning: An Introduction*. A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England.

[Suzuki and Arita, 2000] Suzuki, R. and Arita, T. (2000). Interactions between learning and evolution: The outstanding strategy generated by the baldwin effect. *Proc. of Artificial Life VII*, pages 196–205.

[Suzuki and Arita, 2003] Suzuki, R. and Arita, T. (2003). The baldwin effect revisited: Three steps characterized by the quantitative evolution of phenotypic plasticity. *Proceedings of the Seventh European Conference on Artificial Life (ECAL2003)*, pages 395–404.

[Turney, 1996] Turney, P. (1996). How to shift bias: Lessons from the baldwin effect. *Evolutionary Computation*, 4(3):271–295.

[Watkins, 1989] Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, University of Cambridge, Psychology Department.

[Winter, 1998] Winter, S. (1998). Evolution, categorization and values. *Evolution and Cognition*, 4:185–196.