

University of Edinburgh
Division of Informatics

Pigs and People

4th Year Project Report
Artificial Intelligence and Software Engineering

Jackson Pauls

May 30th, 2001

Abstract: ‘Pigs and people’ is a simulated environment, in which action selection mechanisms can be evaluated and compared. Action selection mechanisms attempt to solve the action selection problem faced by both animals and robots: the problem of selecting which actions to perform in order to achieve heterogeneous and possibly conflicting goals. In this project, two action selection mechanisms are implemented: the non-learning drives mechanism, and W-learning. The non-learning mechanism considerably outperforms the learning mechanism. The results achieved by the two mechanisms are compared and analysed in an attempt to explain the difference in performance.



Acknowledgements

Many thanks to my supervisor, Bridget Hallam, for help and guidance throughout this project. I am also very grateful to Sonia Schulenburg who helped me remain focused early on in the project and understand what I was doing. Mark Ashford, Iona Dias and Robin Parsons provided entertaining discussion and a wealth of ideas for the pigs, the people and the simulated environment. Robin Parsons also created all the images used in the graphical interface.

I have talked about this project with just about everyone I have met in the past year or so, I am extremely grateful for both the attention and the feedback everyone has provided.

Table of Contents

1	Introduction	7
1.1	Motivation	7
1.2	Method	7
1.3	Results	8
1.4	Organisation of this Dissertation	8
2	Action Selection	9
2.1	The Action Selection Problem	9
2.2	Action Selection Mechanisms	10
2.3	Non-learning Mechanisms	11
2.3.1	Distributed Non-Hierarchical Networks	11
2.3.2	An Action Selection Round Up	12
2.3.3	Extended Subsumption Architecture	15
2.4	Learning Mechanisms	15
2.4.1	W-Learning	16
2.4.2	Dynamic Expectancy Model	17
2.4.3	Learning in Hamsterdam	18
2.4.4	And More...	19
3	The Environment	21
3.1	The African Plain	21
3.2	Resources	22
3.3	Predators	23
3.4	Pigs and People	24
3.4.1	Actions	24
3.4.2	Internal Variables	24
3.4.3	The Action Selection Problem Revisited	25
4	Drives for Pig Behaviour	27
4.1	The Drives Action Selection Mechanism	27
4.2	A Pig's Drives	27
4.3	A Pig's Behaviours	28
4.4	Optimising Drives and Initial Results	29
4.4.1	Probability of Changing Direction	29
4.4.2	Impact of Viewed Entities	30
4.4.3	Optimal Drives Results	31
4.4.4	Cause of Death	32
5	W-Learning in People	33
5.1	Learning	33
5.1.1	Reinforcement Learning	33
5.1.2	Q-learning	34
5.1.3	W-learning	35
5.2	The Implementation of W-learning	37
5.2.1	A Person's State-Action Spaces	37
5.2.2	A Person's Reward functions	38
5.2.3	Passing Knowledge Down Generations	39

5.3	Optimising W-Learning and Initial Results	40
5.3.1	Discount Factor	40
5.3.2	Delaying Rate	41
5.3.3	AlphaW and Initial AlphaW	41
5.3.4	W Value Discount Factor	42
5.3.5	Reward Function Scaling Factors	43
5.3.6	Optimal W-Learning Results	44
5.3.7	Causes of Death	45
6	Experiments	47
6.1	Pigs vs. People Apart	47
6.2	Pigs vs. People Together	48
6.2.1	Results	49
6.2.2	Cause of death	50
6.3	A Dry Terrain	51
6.4	Varying the internal state-action space	52
6.5	Variations of W-learning	54
7	Discussion	57
7.1	What was Achieved	57
7.2	What was Discovered	57
7.3	Comparing Drives and W-Learning	57
7.4	The W-Learning Implementation	58
7.4.1	The Internal Stimuli Granularity Parameter	59
7.4.2	Balancing Exploration vs. Exploitation	60
7.5	Further Work	60
7.5.1	Improving the Action Selection Problem	60
7.5.2	New Action Selection Mechanisms	62
7.5.3	Improving this implementation	62
8	Conclusion	65
9	Appendix A: The Code, and How To Run It	67
10	Appendix B: Tables learnt in W-learning	69
11	Bibliography	71

1 Introduction

Action selection mechanisms attempt to solve the action selection problem: the problem of selecting which actions to perform in order to achieve heterogeneous and possibly conflicting goals. ‘Pigs and people’ is a simulated environment, in which action selection mechanisms can be evaluated and compared. This environment was designed and implemented for this project, and two action selection mechanisms were implemented and compared: the drives action selection mechanism, and W-learning. The drives mechanism was inspired by the implementation in [Tyrrell, 93a]; W-learning is introduced by [Humphrys, 96b].

1.1 Motivation

W-learning is a complete, hierarchical action selection mechanism: it specifies not only methods for choosing a behaviour for an agent to perform, but also how these behaviours should choose the low-level actions required to achieve their goals. Although W-learning was already implemented and tested in a simulated environment, its performance had not yet been compared to non-learning action selection mechanisms. In [Humphrys, 96a] W-learning is compared to Q-learning and hierarchical Q-learning, and several possible implementations of W-learning are investigated.

W-learning fared well compared to other learning mechanisms, but the comparison to a non-learning mechanism had not yet been made. ‘Pigs and people’, a simulated environment, was developed to compare W-learning to the drives action selection mechanism, a non-learning mechanism.

1.2 Method

‘Pigs and People’ a simulated environment was implemented, to create an action selection problem suitable for comparing the drives action selection mechanism and W-learning. The drives action selection mechanism was implemented in pigs. W-learning was implemented in people. Both animals can coexist at the same in the environment. Their goal is to survive for as many time steps as possible. This statistic, also referred to as an animal’s ‘age in time steps’, is also used to evaluate that animal’s fitness. Guidelines given in [Tyrrell, 92] were used to create as realistic an action selection problem as possible.

Once implemented, the two action selection mechanisms were first optimised and tested separately, each on its own in the environment. Their performance was then evaluated with pigs and people in the environment at the same time.

The simulated environment was developed in such a way as to allow easy implementation of other action selection mechanisms, thus making it a possible test bed for future investigations. The environment was also implemented in Java and can be run in an applet, thus making it easily accessible to users of all platforms, one aim

being to help create interest in the field of artificial life. Details of how to download and run the code can be found in appendix A.

1.3 Results

The simple drives action selection mechanism fared far better than W-learning in the ‘pigs and people’ simulated environment. Statistics studied are the average and the moving average ages of the species, and the proportion of animals that died of each possible cause of death. Here, the moving average for a species is the average age of the last hundred individuals of that species at their time of death.

In all experiments, the pigs outperformed the people by a large margin. In a typical run, after 100000 time steps, the pigs’ average age is 1294 time steps, living over seven times longer than the people’s 171 time steps. Possible reasons for this are investigated. Additionally guidelines are provided on further work required to further compare these two action selection mechanisms.

1.4 Organisation of this Dissertation

In Section 2, we introduce the action selection problem, which is typically encoded in an environment, and is what action selection mechanisms attempt to solve. A review is also given of some learning and non-learning mechanisms that have been proposed over the years. Section 3 gives the details of the ‘pigs and people’ environment and the action selection problem it poses to its two main characters: the pigs and the people.

The drives action selection mechanism is introduced in section 4, which was implemented in the pigs in the simulation. The drives mechanism is also optimised and evaluated, to see how well the pigs fare without people in the environment. Section 5 details the workings of W-learning, which was implemented in the people. Preliminary results are given, which were used to find appropriate parameters for this action selection mechanism.

Experimental results under a variety of environmental conditions are given in Section 6, and reasons for the different conditions are suggested. Section 7 compares the implementation of both action selection mechanisms, and discusses further work possibilities. Finally, Section 8 concludes this dissertation.

2 Action Selection

Action selection, as a research topic, is new even by artificial intelligence's standards; it has been actively studied within the field only since the late 1980s. The interest derives mostly from the need to build efficient robot controllers, but also partly from the desire to model animal behaviour. However, in the latter case, the interest is mainly generated from outside the field of artificial intelligence. Tyrrell gives the most complete and up to date definition of the problem [Tyrrell, 92]; but earlier work in artificial intelligence can be related to action selection: robots have always needed control mechanisms to tell them what to do at each given point in time. However, the task performed by the controllers was usually more akin to planning than action selection.

We will introduce non-learning action selection mechanisms, which were the main focus of research until around 1993, at which point Toby Tyrrell finished a comprehensive comparison of the mechanisms described to date in his PhD thesis [Tyrrell, 93a]. Then we shall look at a variety of different architectures for learning mechanisms, some of which are still actively researched and experimented with today. However, let us first describe the problem that these mechanisms attempt to solve: the action selection problem.

2.1 The Action Selection Problem

The action selection problem arises when any agent that can perform a variety of actions attempts to satisfy one or several goals in an environment. [Tyrrell, 93a] gives a very neat definition: 'the problem of action selection is that of choosing at each moment in time the most appropriate action out of a repertoire of possible actions'. Fortunately, there is no real dispute over the definition of the problem. However, we need to be more explicit about the purpose of the actions stated above.

The purpose of a selected action would of course be to satisfy one of the agent's goals such as obtaining and eating food; but the agent may have several conflicting and heterogeneous goals with different levels of urgency, affected by both internal and external stimuli. Actions are usually grouped into behaviours; for example, the actions 'look around' and 'move in direction D' may be components of the behaviour 'stay out of people's way'. The agent may have any number of different behaviours, depending on its complexity; and any one action may be useful for any number of different behaviours.

The agent may need to be opportunistic, and take advantage of its current situation by, for example, consuming a resource when it is available. It may also be the case that a particular action partially satisfies more than one goal, for example moving in a certain direction may help the agent both avoid a Bad Thing, and also approach a Good Thing. These are a few aspects of the action selection problem; there may be many still undefined. This is what makes the problem interesting, but also, necessarily, difficult. Tyrrell [1992] identified different dimensions of sub-problems

that should be present in an animal's action selection problem; these are introduced in section 3.4.3.

For **example**, let us consider a mouse as an agent, living in some unnamed building of an unnamed university. The mouse will have several goals, a subset of these could be: obtaining food, avoiding people, remaining as close as possible to its mouse hole, and keeping its fur free of parasites. When the mouse has infiltrated Dr. X's drawer, in which a sandwich is located, a sensible behaviour would be to consume the food, and this could be considered a simple display of opportunism. A less sensible option would be to clean its fur. When Dr. X opens his drawer, the mouse may try to satisfy two goals at once by running away, thus partially satisfying his goal of avoiding people, but also run in a specific direction, to partially satisfy its goal of remaining near its mouse hole.

The **usefulness**, from a roboticist's point of view, arises from the need for efficient robot controllers. [Möller et al., 00] suggests that the study of animal behavioural mechanisms can help us build better robot controllers. Work in this field can be related both ways: robots that simulate animal behaviour may help validate hypotheses about the inner workings of the animals, and studying the inner workings of the animals may help the development of multi-functional robots that could operate in the real world.

Additionally, much can be said in the study of cognition, if realistic models of behaviour can be implemented using action selection mechanisms in a realistic simulated environment.

2.2 Action Selection Mechanisms

Action selection mechanisms are methods, algorithms... any sort of mechanism that describes a possible solution to the action selection problem. We will give an overview of a variety of these mechanisms, developed by a number of researchers.

The initial drive to study these mechanisms came from the field of robotics, where more and more complex robots were being developed. The mechanisms used to control the robots needed to be able to cope with a new variety of actions they could perform, in order to accomplish an increasing number of goals. As the complexity of the tasks one could expect from a robot increased, the problem of choosing which action was most appropriate at a given time for the robots became increasingly similar to that same problem for living creatures.

Many approaches have been proposed to the implementation of effective robot controllers. Most lie somewhere in between pure planning and behaviour based paradigms. The *planning* approach attempts to create an accurate model of the world, on which inference is performed to evaluate what action to take at each point in time. An example of a typical planning system, STRIPS, can be found in [Fikes and Nilsson, 71]. Although successful in restricted environments, the planning paradigm is generally found not to be reactive enough for a dynamic and unpredictable environment. The pure *behaviour based* paradigm does not use a model, but reacts to the state of the environment currently being sensed using what can intuitively be seen

as instincts. The subsumption architecture is an example of an implementation of behaviour based design [Brooks, 86]. Behaviour based systems typically perform well on a low-level, reacting adequately to the immediate environment despite its unpredictability. However, they lack the cognition to perform high-level complex tasks.

What seems to be needed is an approach somewhere in between the planning and behaviour based extremes. Furthermore, such a mechanism could be inspired from animal control theories, or even integrate a form of learning, as we shall see in section 2.4. As the study of animal behaviour and how their behaviours are achieved may help determine efficient action selection mechanisms for robots, we shall encounter situations where the main agent is described as an animal. In other cases, the focus of the study will be a simulated robot.

2.3 Non-learning Mechanisms

Non-learning action selection mechanisms were the first to be investigated, in order to explain animal behaviour or as a simple implementation to achieve the required behaviour from a robot. Following is a tour of some of the more influential mechanisms proposed; importantly, all were evaluated and compared within the same simulated environment.

2.3.1 Distributed Non-Hierarchical Networks

Maes was one of the first people to specify a new action selection mechanism from within the field of artificial intelligence. In [Maes, 90] a distributed non-hierarchical network is specified as an action selection mechanism. Her ‘spreading activation’ system combines characteristics from both traditional planners and reactive systems, using connectionist computing and low-level distribution, in an attempt to create a robust mechanism capable of coping with a dynamic and unpredictable environment.

The mechanism is based on a number of ‘competence modules’, which from now on we will call behaviours for consistency across action selection mechanisms. Behaviours in this system resemble operators in traditional planning, having a set of preconditions, as well as an ‘add’ list and a ‘delete’ list. Furthermore, all behaviours have a level of activation, and are linked by successor, predecessor and conflict links, in order to form a network. Binary sensors are used to get information about the environment. At each time step, goals, internal and external stimuli are fed into this network, the activation level for each behaviour is calculated based on the links in the network, and the behaviour with the highest activation is selected as the most appropriate at that time.

This mechanism was tested with a simple agent, a robot with two hands, and a set task for the robot: sand a board, and then paint itself. This mechanism was also experimented with further, in a more interesting simulated environment containing animal agents, food, water and obstacles [Maes, 91]. Performance of the mechanism was quite good, and very encouraging at the time. Interesting properties from both an ethological and a robotics point of view could be observed, such as ‘Goal-

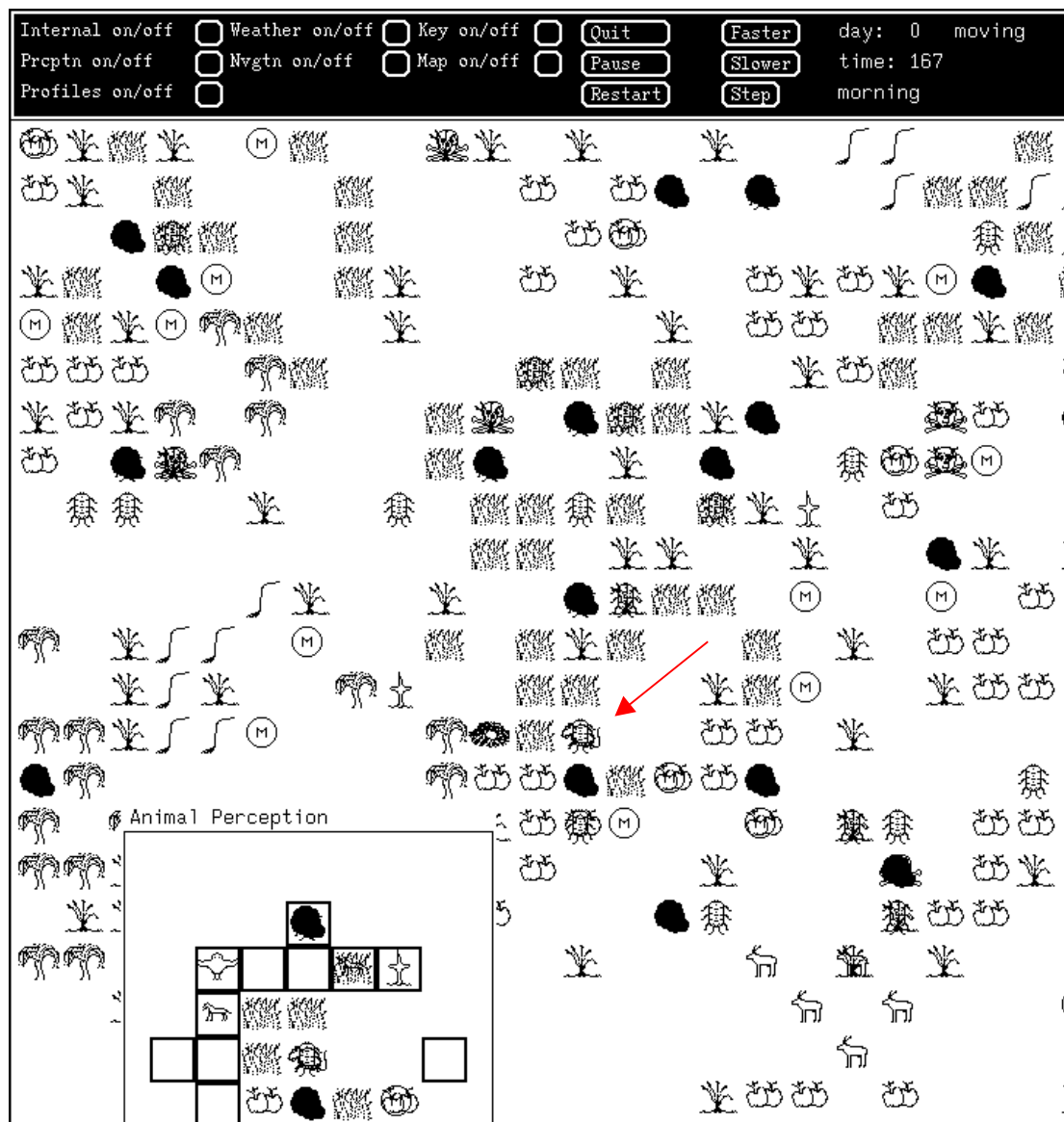
Orientedness’, ‘Varying Attention’ and ‘Irrelevant Behaviour’. Furthermore, the agent’s behaviour could be modified with the setting of parameters; for example, a manually set threshold could vary the trade-off between ‘Speed’ and ‘Thoughtfulness’. However there are a large number of such global parameters in this system (for thresholds, link activation ‘energies’ among others), and each of these has to be tweaked by hand. As we shall see this is a recurring issue in action selection mechanisms.

2.3.2 An Action Selection Round Up

Tyrrell, rather than propose a new action selection mechanism of his own, compares seven ‘computational mechanisms’ for action selection in [Tyrrell, 93a]. Most of the mechanisms he examines come from outside the field of artificial intelligence, and many did not have a complete specification. However, he manages to implement five different action selection mechanisms in a complex simulated environment, based on the habitat of a small mammal. Maes’ mechanism is one of those implemented, along with a simple drives model, a mechanism proposed by Lorenz [1981], a strict hierarchical decision structure inspired by those of Tinbergen [1951] and Baerends [1976], and Rosenblatt and Payton’s mechanism [1989].

The environment Tyrrell implemented is worth looking at in a bit more detail, as it is the closest to the environment implemented for pigs and people. It is also the most complete and interesting of all those implemented, at least among those used to evaluate the action selection mechanisms described in this paper. The character of interest in Tyrrell’s environment is a small animal, which has a den where it must spend the night. During the day, this animal must satisfy its need for water and for several different types of food, as well as keep its fur parasite-free. It must avoid predators and large animals that could crush it as well as avoiding toxic sources of food and water and dangerous places. It must also attempt to mate when possible, recognise where it is, and remember where it has been thanks to landmarks in the environment. The animal’s genetic fitness is evaluated by counting how many times it successfully mates during its lifetime. Furthermore, Tyrrell developed a graphical interface for this simulation, so that the happenings could be easily and comfortably monitored whilst the simulation was running (see screenshot on next page).

Unfortunately, the graphical interface only runs on machines that support a now-obsolete window toolkit; if this were not the case, Tyrrell’s simulated environment would be an excellent test bed for any newly proposed action selection mechanisms. However, Bryson [2000] integrated her mechanism into Tyrrell’s environment, which fortunately can also be run without the graphical interface, to compare it with the best of the already implemented mechanisms, as discussed in section 2.3.3.



A screenshot from Tyrrell's Environment. The agent, a small rodent, is on the same square as a bug-like creature (prey) near the centre of the image.

Figure 1 is a summary of the results obtained by Tyrrell in his final implementation of all tested mechanisms. The fitness is simply the number of times the animal successfully mated; higher numbers are better.

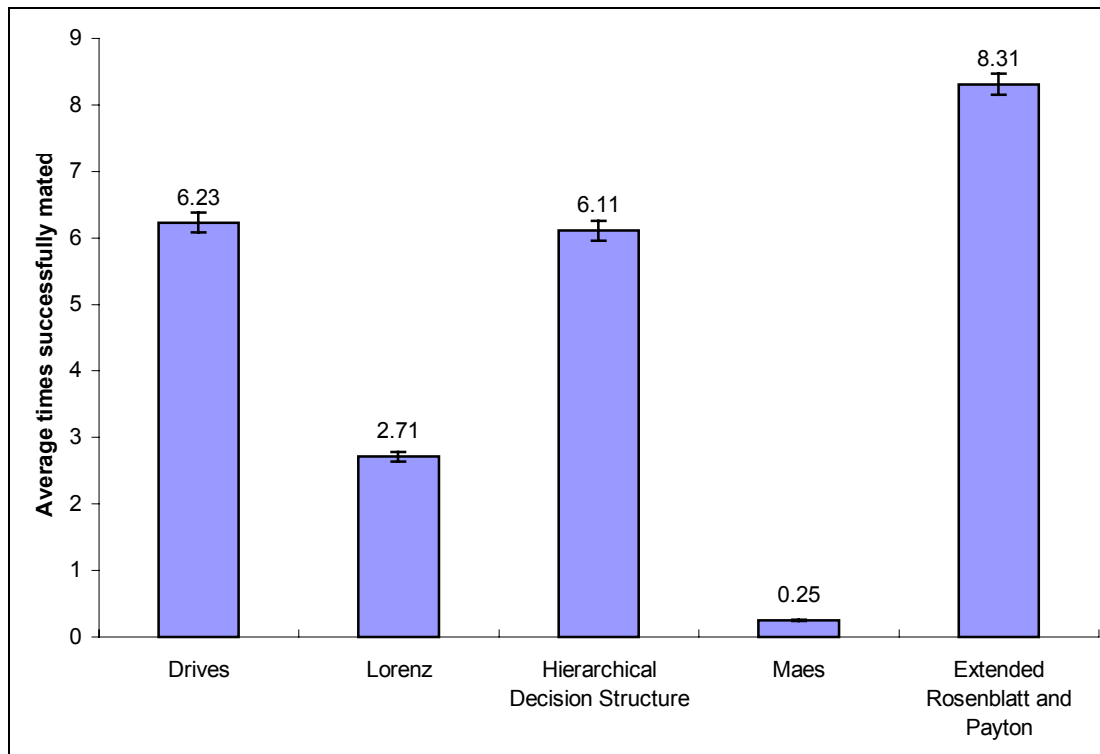


Figure 1: Performance of the different action selection mechanisms implemented in Tyrrell's environment. The T-shaped intervals represent the standard deviation.

The mechanism that fared the best in the experiments run by Tyrrell was a version of Rosenblatt and Payton's mechanism, which is based on a hierarchical, connectionist, feed-forward network [Rosenblatt and Payton, 89]. To obtain these results, Tyrrell extended this mechanism by adding temporal and uncertainty penalties to actions that either would take too long before a reward was obtained, or for which it was uncertain that a reward should be obtained at all.

The action selection mechanism that performed the worst was Maes'. There are various issues that may have caused Maes' mechanism as described in section 2.3.1 to perform poorly in the system developed by Tyrrell. A lack of hierarchy in the mechanism could be the main problem; loss of information due to binary sensors, no allowance for partial satisfaction of a goal and lack of persistence are a few others stated by Tyrrell. Tyrrell [1994] gives a more detailed evaluation of Maes' mechanism.

Based on his experiments, Tyrrell identified a number of desirable properties for action selection mechanisms, the main one being the need for a free-flow hierarchy at the heart of the mechanism. A free-flow hierarchy is one that places no restrictions on the flow of information or activation through the hierarchy, and this allows the combination of preferences from high-level layers to low-level layers in the network. The need for hierarchy can be justified by the fact that it is likely to be similar to what real animals use, based on the work of Baerends [1976]. There is also the fact that Rosenblatt and Payton's hierarchical mechanism performs better than the non-hierarchical mechanisms in Tyrrell's simulation. A more detailed investigation into the use of hierarchies is reported in [Tyrrell, 93b]. Based on his findings, Tyrrell

gives a solid platform for further research into more computational methods for action selection.

Both strict hierarchy and drives action selection mechanisms performed well. As the drives is also a simple mechanism, it was chosen for the pigs in ‘pigs and people’. The drives mechanism is described in section 4.

2.3.3 Extended Subsumption Architecture

Bryson introduces an action selection mechanism named Edmund [Bryson and McGonigle, 97] that uses object oriented programming as an effective tool to implement an extension of the subsumption architecture proposed by Brooks [1986]. The main components of this architecture are behaviours. Edmund’s behaviours are akin to subsumption architecture behaviours, except for the fact that Edmund’s behaviours are extended to have a long-term memory, and can have more interaction with other behaviours within the system. Indeed, behaviours in Edmund are allowed to both initiate another behaviour, and get information about the state of another behaviour. Furthermore, although Edmund adheres to the bottom-up paradigm of the subsumption architecture, Edmund does not assume that existing behaviours are correct when implementing a new behaviour. Edmund also attempts to combine the advantages of free flow hierarchical control, to keep the problem manageable, and explicit sequencing, to encode fixed action patterns, which can be followed over several time steps.

In [Bryson, 00], Edmund was compared to the extended Rosenblatt and Payton architecture implemented by Tyrrell. Edmund was implemented in the small animal in Tyrrell’s environment, and a large range of tests was performed. Edmund outperformed the extended Rosenblatt and Payton architecture by a small factor, and was also found to be more portable. Not only did Edmund’s outperformance become larger on a new environment with scarce resources, but the final implementation of Edmund was also far simpler than Tyrrell’s implementation of the extended Rosenblatt and Payton architecture.

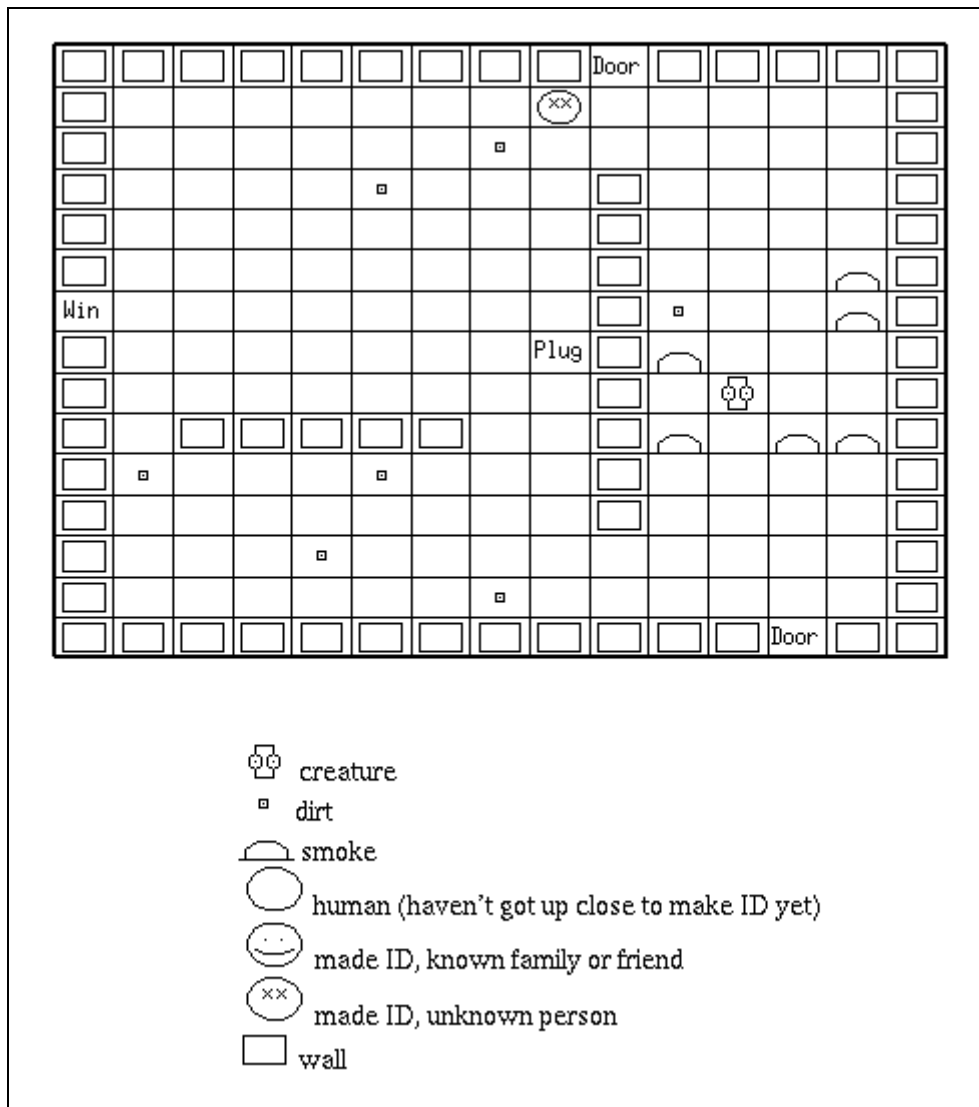
2.4 Learning Mechanisms

Learning Mechanisms were the predominant focus of artificial intelligence research in the second half of the 1990s, and are still actively researched today. A major criticism of the computational mechanisms implemented by Tyrrell, and all other non-learning mechanisms is the amount of hand tuning and tweaking of parameters necessary to adapt a mechanism to an agent and its environment. It has been in fact noted that ideally the fitness function for an agent should be implicit in the environment. Learning mechanisms can help minimize the amount of manual tuning, by allowing the agent to learn its behaviour from the environment, or from a reward function, thus setting its own parameters. Furthermore, non-learning mechanisms require the implementer of the mechanism to have knowledge of the problem faced by the agent, which may not always be a realistic assumption.

There were various concurrent efforts aiming to introduce reinforcement learning (see section 5) into action selection mechanisms, notably by Humphrys [1996], Witkowski [1998] and Blumberg, Todd and Maes [1996]. These three mechanisms were developed from completely different backgrounds, and the different approaches and solutions will be introduced. The terminology can get a bit confusing, as often ‘agents’ are split up into different components also called ‘agents’. Here, the robot or animal will be called the ‘agent’; any components within the agent will be called ‘sub-agents’ or ‘behaviours’.

2.4.1 W-Learning

Humphrys [1996a] compares several variants of both Q-learning and W-learning. These methods are devised almost directly from reinforcement learning: Q-learning was adapted from Christopher Watkins’ work on ‘Learning from Delayed Rewards’ [Watkins, 89], and W-learning is a variant of this developed by Humphrys himself. Humphrys tests these learning action selection mechanisms in a simple simulated environment, in which a house robot had to perform a variety of tasks, such as cleaning up dirt after people, putting out fires and detecting intruders, whilst maintaining its power levels by plugging itself into a power socket when necessary (see screenshot on next page).



The house robot simulated environment, and its legend

The simulated environment used by Humphrys prevented these mechanisms from being tested to their full capabilities, the rewards for opportunism being too low. However, the performance in his environment was very encouraging, and thus W-learning was the action selection mechanism implemented for the people in the 'pigs and people' simulated environment and is discussed in detail in section 5.1.3.

2.4.2 Dynamic Expectancy Model

Witkowski [98] proposes an action selection mechanism that is inspired by and improves upon Drescher's 'intermediate level cognitive' model [Drescher, 91], Mott's 'sensori-motor' model [Mott, 81], and Roitblat's 'cognitive action theory' approach [Roitblat, 91]. Witkowski calls his mechanism the Dynamic Expectancy Model. Schemas, a three-part representation of knowledge, are key to this approach. Schemas

are defined as ‘context-action-outcome’ triples, called μ -hypotheses in his paper. These μ -hypotheses are what are learnt in Witkowski’s system.

Witkowski’s mechanism is quite different from the others introduced, as the agent’s goals are specified externally: the agent is called upon to perform a specific task at certain times during execution by an outside observer. When not being asked to perform a task, the agent attempts to build new μ -hypotheses, or reinforce existing ones. His method also adopts a predictive view, in that changes to learnt information can be made based simply on whether a predicted event occurred or not. This is what the learning in his system is based on. These predictions are used for learning in two ways: to confirm existing μ -hypotheses if a prediction is accurate, and to detect unexpected events in order to trigger the creation of new μ -hypotheses if the prediction was off-target.

The actual action selection is split into two categories: goal directed and exploratory. When in exploratory mode the agent performs random actions, attempting to confirm existing μ -hypotheses and create new ones. When in goal directed mode, μ -hypotheses are chained together by the agent to form ‘dynamic policy maps’, this process being referred to as ‘spreading activation’ (not to be confused with Maes’ mechanism with the same name). These dynamic policy maps indicate which action to perform next, given the current state: this will be the first action in the chain that is predicted to eventually accomplish the given goal, or at least partially satisfy it.

This method is still being developed. Although it has been tested in a simple simulated environment, to prove that the concepts work, it has yet to prove itself in an environment as complex as Tyrrell’s or even Humphrys’ where the action selection problem is a much harder one. One may fear that the lack of hierarchy in this mechanism and its complexity may be severe obstacles in a more realistic environment.

2.4.3 Learning in Hamsterdam

Blumberg, Todd and Maes [96] build on an action selection architecture proposed in [Blumberg, 94] with which they integrate certain types of associative learning. Sutton’s Temporal Difference learning algorithm [Sutton and Barto, 90], a form of reinforcement learning, was used as the learning component in this action selection mechanism. Hamsterdam is a toolkit for building autonomous animated creatures. Creatures in this system have three basic characteristics: geometry, motor skills and a behaviour system. There are two layers of abstraction between these three parts: the controller and the degrees of freedom. The purpose of the behaviour system, of which the action selection mechanism is part, is to send the right set of controls to the motor system at every time-step.

The structure of the behaviour system is a distributed network of self-interested, goal directed entities; from now on, these will be simply called behaviours. ‘Releasing mechanisms’ determine the appropriateness of each possible behaviour, and output a continuous value dependent on: the presence of a stimulus, that stimulus’s features, the distance of that stimulus from some ideal distance. This

value can then be combined with the motivational strength, which comes from internal variables such as hunger. The reinforcement learning that takes place on these releasing mechanisms, is of:

- New contexts for goal satisfying with existing behaviours,
- New releasing mechanisms for consummatory behaviours,
- New releasing mechanisms with greater predictive power.

Therefore, what are effectively learnt are different types of associations between contexts, behaviours and outcomes. Learning was thus successfully integrated in the action selection mechanism proposed by Blumberg. However, there may have been other ways to integrate learning, the implemented one being similar to animal conditioning.

Again, this mechanism requires much more testing, as is noted by the authors. However, the testing that was performed on both a virtual dog and a virtual hamster shows a certain potential in the method, as the agents were able to learn on a variety of different levels.

2.4.4 And More...

There have been many other approaches taken to solve the action selection problem using learning, for example [Sutton, 90], again using reinforcement learning. In Sutton's mechanism, a single process executes alternately on a model of the world, and the real world. He introduces '**Dyna architectures**' which combine learning and planning into a single process, and tests his mechanism in a simple environment.

Additionally, [Spier, 97] and [Spier and McFarland, 97] use *drk*, a **reactive motivational model** as an action selection mechanism, and extend it with reinforcement learning. This work is especially interesting from a biological point of view, as it focuses on reproducing behaviours observable in real animals.

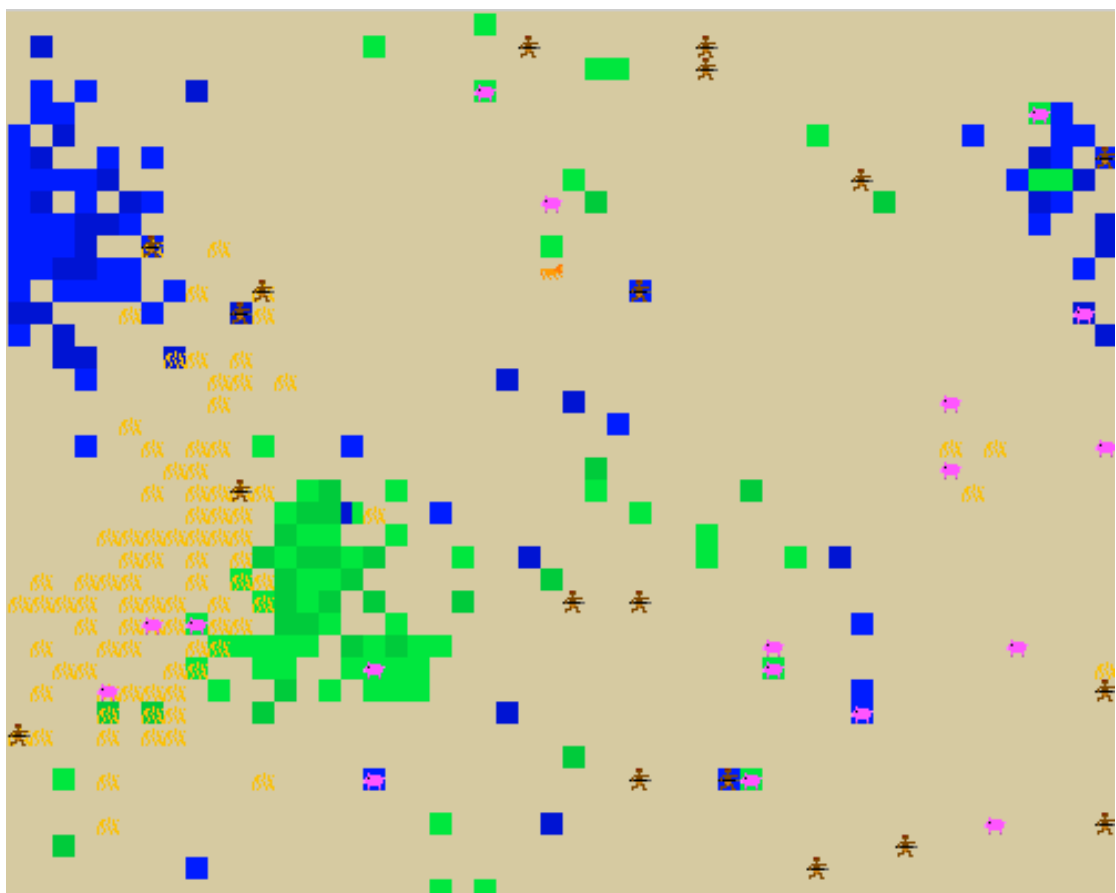
We can expect more mechanisms to appear, as learning in action selection mechanisms is far from mastered, and is an inspirational topic of research. Any new mechanism should obviously build on knowledge gained from existing mechanisms, such as those just described.

3 The Environment

This section describes the environment developed as part of this project. The setting of ‘pigs and people’ is an African plain, where both slop and wheat grow, and rain falls and collects in large puddles. Pigs, people, and lions coexist in the plain.

3.1 The African Plain

The setting is an African plain, with water, slop and wheat as the available resources; pigs and people that feed on these; and lions that feed on the pigs and people. This is a typical screenshot of the environment, from running the ‘pigs and people’ applet, containing 20 pigs, 20 people and 1 lion.



Screenshot of the pigs and people applet.

The environment is a grid world. The grid’s dimensions are adjustable parameters of the application, however, all experiments described in this paper took place on a 50 by 40 grid like the one figured above. The environment

wraps around on all four sides, and would thus look like a torus in three-dimensional space.

At each time step in the environment, first all the pigs, and then all the people and finally the lions get to perform an action of their choice. This ordering does not introduce a bias: over several time steps, it becomes a cycle. Which animal performs the first action in a time step is then irrelevant. There are 16 hours of daylight, and 8 of night. By day, animals can see as far as their `vision_radius` parameter allows; during the night, all animals' vision is restricted to one square all directions.

3.2 Resources




The environment contains three basic resources: water, wheat and slop. Water is a resource essential for both pigs and people. Only pigs need slop and only people need wheat. The total number of entities of each resource remains constant over time, when one entity is fully consumed, then a new one appears on the terrain.

Each resource entity has a variable (`amount_left`), which represents how many resource units are available from that entity. For example, if a water entity has `amount_left` = 13, then any animal that wishes to drink from that entity can decrease its internal thirst variable by a maximum 13; and would thus consume that entity entirely, causing it to be removed from the square it was on. When a new entity is placed on the terrain, its `amount_left` is assigned a uniformly distributed random integer value between 1 and 100 inclusive.

The number of entities of each resource that are initially placed on the terrain can be set in the parameters. Each entity is placed at a random walk of length 5 from a randomly selected entity of the same type with a probability of (`proximity_probability`), which can be set individually for each type of resource. Otherwise, the entity is placed on a random square in the grid. No two resources of the same type can coexist on the same square, but any combination of one water entity, one wheat entity and one slop entity can share the same square.


This mechanism for placing resources ensures that the resources are placed close to other resources of the same type to start with. As each resource is consumed, the individual entities may group to form large clusters, making the problem more difficult for the animals needing to consume these resources, as they become less evenly spread around the terrain. These large clusters may move around the terrain slowly as they are consumed, and may eventually break up, due to the randomness of the placement mechanism. This was considered realistic as resources are typically found in clusters. The `proximity_probability` parameter ensures that a percentage of each type of resource is sparsely placed around the whole terrain. Unless specified otherwise, the following values were used in the simulation:

- The number of entities of each resource was set to 90.
- The `proximity_probability` parameter for each resource was set to 0.6.

Water is represented as a blue square  in the applet version of the environment, and slop is represented as a green square . This icon represents wheat: . Both water and slop may be present on the same square in the grid; in this case, they are represented as two rectangles, a blue one on the left of the grid cell for the water, and a green one on the right for the slop. The wheat image has transparent pixels, and is drawn above the other two resources. Thus, it is possible to visually identify exactly which resources are on a given square.

Pigs and people are the necessary resource for the lion; however, our point of view is that of the pigs and people as they are the main characters in this project, and thus the lion will always be called the predator (section 3.3).

3.3 Predators

Lions, represented in the applet by , actively hunt pigs and people. Lions are only awake during the day, and have a very simple hand-coded behaviour. If a lion can see a pig or a person, it will chase it for a while, then get tired and wait until the fatigue passes. If a lion cannot see any prey, it will move randomly until it spots one. If a lion can see two prey animals, it will chase one of them at random. A lion catches a prey by moving onto the same square as a prey, that prey then dies and is considered eaten. The lion sleeps during the night, and is then not a threat to anything. Lions are immortal, and the same lions are present from the beginning to the end of the simulation. This simplification is justified by the fact that lions were only implemented as an occasional and urgent stimulus for both pigs and people.

The initial number of lions in the environment (`initial_lion_count`), the number of time steps a lion will chase a pig for (`lion_determination_max`), the number of time steps a lion will wait before moving again (`lion_wait_max`), the probability that a lion will go in a new direction when making a random move (`lion_direction_change`) and the number of squares a lion can see in all directions (`lion_vision_radius`) are parameters in the simulation. In the experiments described later, the following values were set unless specified otherwise:

- `initial_lion_count` = 1 (only one lion)
- `lion_determination_max` = 10 timesteps
- `lion_wait_max` = 10 timesteps
- `lion_direction_change` = 0.3 (30%)
- `lion_vision_radius` = 4 squares

3.4 Pigs and People

Pigs 🐷 and people 🧑 consume the resources provided by the environment, and are in turn consumed by lions. They share the same set of internal variables and action repertoire. Indeed, to ensure this, the two classes representing both animals share the same code. The initial number of pigs and people in the terrain, the probability that the animal will go in a new direction when making a random move (although this doesn't apply to people as they never select the random move action) and the number of squares they can see in all directions (3) are parameters to the simulation. Pigs and people can also be active during the night, but their vision radius is reduced to one during this time.

3.4.1 Actions

At each time step in the simulation, all pigs and people get the chance to perform one action. Possible actions are drink, eat, relax, move, random move, and sleep. Although the animals choose which action to perform, the result of that action depends entirely on current state of the environment. Thus, a person can choose to drink even if he is nowhere near any water, and must learn of his own accord that there is no benefit in doing so. As we shall see, in the case of the drives mechanism implemented for the pigs, this knowledge resides in the designer, and is implicitly hand-coded within the mechanism.

This choice of actions is the basis of the interaction between the pigs and people and the environment. If a person drinks all the water on a square, that square is then dry; but water may again appear on it at some later stage. The same goes for the other two resources: slop and wheat. The actions have an effect not only on the environment, but also on the animal's internal variables. See section 3.4.2 for a complete description of this.

The move and random move actions have certain particularities. An animal can move in eight possible directions: up, down, left, right and the four diagonals in between. However, a move action also represents the fact that an animal wants to do nothing, ie to move in a 'null direction'. Thus, an animal's internal variables are changed differently when it moves in a direction to when it moves to the null direction, as an animal will get less tired from standing still than from moving about. Indeed one of the first things people learned to do was stay still and live forever, taking advantage of a bug that caused their internal variables to remain unchanged as long as they always stood still. This was fixed early on in the implementation.

3.4.2 Internal Variables

Pigs and people have internal variables for thirst, hunger, heat, fear, fatigue and sleep. These are all initialised to zero; and all except fear have a fatal level. If an

internal variable goes beyond its fatal level, then the animal dies. All other internal variables vary between 0 and 100, the animal dying if any reach 100.

Internal variables are incremented and decremented depending on the animal's actions:

Actions / Variables	Thirst	Hunger	Heat	Fatigue	Sleep
Move / Random Move	+1	+1	+1	+5	+3
Drink	-A/+1	+1	-1/+1	+1	+3
Eat	+1	-A/+1	+1	+1	+3
Relax	+1	+1	-10	-20	+3
Sleep	+1	+1	-5	-15	-20
Nothing	+1	+1	+1	+1	+3

Where A is the amount consumed, and 'Nothing' is the action performed when an animal attempts to consume a resource currently unavailable from the square it is on, or when the animal stays still by moving in the null direction. The odd one out is the fear variable that increases suddenly if a lion can be seen, and decreases with time if no lion is longer visible. Fear cannot be fatal, but it is used to calculate the fear drive.

No variable can be negative, and is set to zero if the above calculations would make it negative. The amount A, drunk when the drink action is selected, depends on the animal's thirst level and the amount of water available: if animal's thirst level is 7, it can drink at most 7 units of water. If there are only 3 units available on the square it is currently on, 3 is the maximum it can drink in that time step. If there is no water available, then the thirst level increases by one and so does the heat level. The same is applied for when the animal selects the eat action. The most an animal can eat and drink at any time step is set in the animal superclass and is therefore the same for both pigs and people, and is set to 30 units unless specified otherwise in this document.

3.4.3 The Action Selection Problem Revisited

Pigs and people face the action selection problem created by their internal variables and the environment. Their main objective is to survive as long as possible. This is how their fitness is evaluated. [Tyrrell, 92] identified a diversity of types of sub-problems, which need to be considered to ensure the scope of an action selection mechanism is suitable, compared with that of a real animal. Each sub-problem fits somewhere along each axis of a 7-dimensional space: using examples from the pigs and people environment, the sub-problems he identified fall along the following axes:

- **Homeostatic to non-homeostatic:** maintaining food, water, fatigue, heat and sleep levels are all homeostatic problems. Escaping predators is mostly non-homeostatic, even though it can be considered as the problem of keeping the internal fear level down to an adequate level.

- **External stimulus dependent to external stimulus independent:** obtaining food, water, escaping predators are mostly dependent on external stimuli; maintaining appropriate levels of fatigue, heat and sleep are independent of external stimuli.
- **Internal stimulus dependent to internal stimulus independent:** maintaining appropriate levels of fatigue, food, water, heat and sleep are internal stimulus dependent. Escaping predators is internal stimulus independent.
- **Periodic to non-periodic:** maintaining the sleep level is essentially a periodic problem, maintaining food, water, fatigue and heat levels, and escaping predators are essentially all non-periodic.
- **Continual to occasional:** maintaining food, water, fatigue, heat and sleep levels are all continual problems, escaping predators is however obviously occasional.
- **Urgent to non-urgent:** Escaping predators is the only problem that is always urgent. The urgency of other problems depends on the current internal stimuli of the animal: if the animal is not thirsty, then obtaining water is a non-urgent problem. However the problem becomes increasingly urgent as the internal thirst level approaches its fatal limit.
- **Prescriptive to proscriptive:** All problems in pigs and people are prescriptive, ie maintaining food, water, fatigue, heat and sleep levels and escaping predators all prescribe a unique action to take. A proscriptive problem would present the animal with one or several actions *not* to take, and leave the choice of action somewhat more open.

Note that the overall problem of obtaining and consuming a resource is dependent on both internal (thirst or hunger level) and external stimuli (whether the appropriate type of resource can currently be seen). The issues of not having a proscriptive sub-problem are discussed in section 7.5.1.

4 Drives for Pig Behaviour

The drives action selection mechanism is very simple, yet has been proven to work quite well in Tyrrell's environment (see section 2.3.2), which is similar to the pigs and people environment. This is why the drives mechanism was chosen for the pigs in 'pigs and people'.

4.1 The Drives Action Selection Mechanism

The drives mechanism used for the pigs is directly inspired from that used for the small animal in [Tyrrell, 93a], which is in turn adapted from [Hull, 43]. A *drive* is the total motivation, combined from all stimuli, for a given behaviour. A *behaviour* attempts to satisfy one of the animal's goals, by selecting the appropriate low-level actions to take, given the current perceived state of the environment. Drive strengths are calculated for every possible behaviour; the behaviour with the highest drive is the one that is allowed to select which low-level action is to be performed. If two behaviours share the same highest drive, then the behaviour chosen is based on the manually selected order: run away, drink, eat, cool down, sleep and finally rest. Thus, if run away and sleep have then same drive, the run away behaviour would get to select which low-level action to perform.

The drives action selection mechanism does not itself determine which low-level action to perform once a behaviour has been selected. This must be decided in some other manner. The way this is decided for the pigs is detailed in section 4.3. Here, a drive is a function of only the internal and external stimuli; however, if there was any learning in the pig's mechanism, experience could also be incorporated as a variable in the drive calculations.

The drive functions and behaviours need to be hand coded, but optimal coding may not always be obvious. Indeed, considerable experimentation was required in order to achieve appropriate pig behaviour. The main issue encountered was balancing the drive calculations (see section 4.2): each behaviour must have a chance to perform its actions. We shall now detail the implementation used throughout the experiments in this paper.

4.2 A Pig's Drives

The following is a plot of how a pig's drives vary depending on its internal variables.

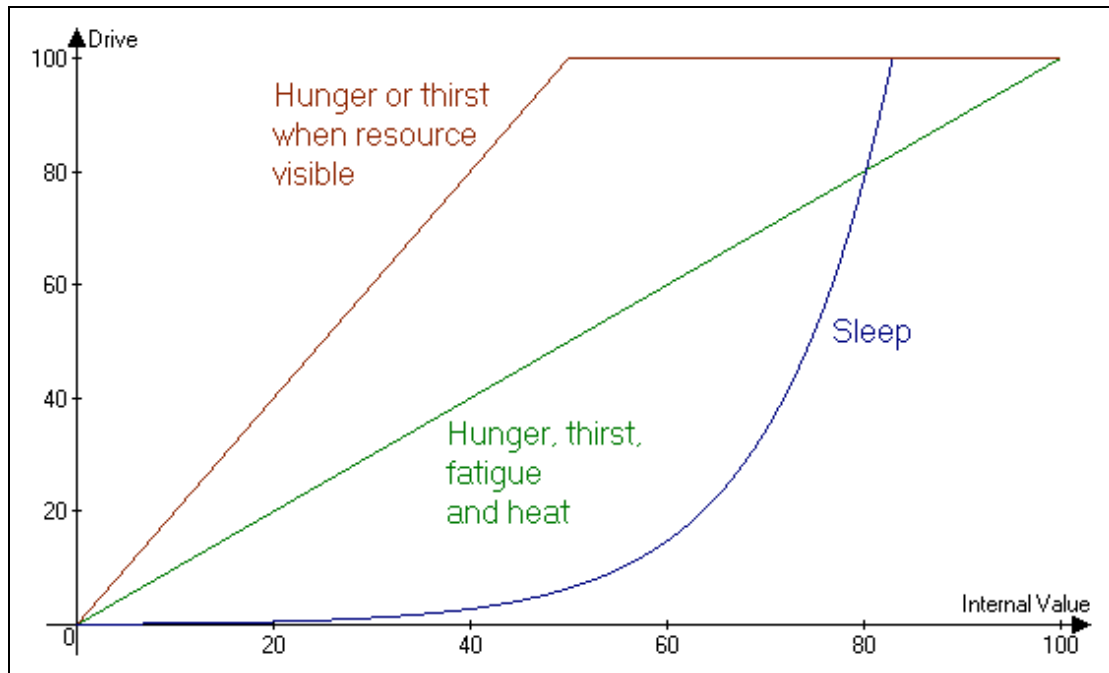


Figure 2: The drive functions used in the pigs' drives action selection mechanism.

External stimuli are the visibility of slop, water and lions. In the graph above, only the calculations for the 'eat' and 'drink' behaviours take account of external stimuli. However, the run away behaviour is also dependent on the external stimulus of spotting a lion. In fact, the drive calculation for the fear drive is not figured above, as it is almost purely dependent on external stimuli, if a pig can see a lion, his internal fear level jumps to 95, and this decays by 20 at each time step thereafter when the pig cannot see a lion.

The slope of the red line above, for when a resource is visible, varies, depending on a parameter (`impact_of_viewed_entities`). The `impact_of_viewed_entities` parameter is an attempt to represent how much the fact that a pig can see a type of resource should impact its drive to consume that type of resource. This is set to 1 in the above graph, and is optimised in section 4.4.2.

4.3 A Pig's Behaviours

Behaviours attempt to solve pigs' sub-problems by selecting the appropriate low-level actions to perform at every time step. A pig can have the following behaviours: eat, drink, cool down, run away, rest and sleep. In this section we describe what low-level actions each behaviour chooses to perform, if it is selected as the behaviour with the highest drive.

Both the eat and the drink behaviours are concerned with resource consumption, and have the following implementation:

- If a pig is on the same square as a resource and has a deficit in that resource, it consumes that resource.

- If a pig can see an entity of the required resource, but is not on the same square, it moves towards that perceived entity. If there are several entities of the required type, the pig chooses the closest one.
- Otherwise, it explores with a random movement.

When performing a random movement, the direction to go in is initially set randomly. A parameter, `pig_direction_change`, determines the probability of a new direction being selected, when making a new random move; otherwise, the last random direction is used, which is stored for each pig. The last direction, random or not could also be used: the implementation was chosen arbitrarily. The exception to this rule is when the last random direction chosen was the null direction, causing the pig to remain in place: a new random direction is selected in this case.

The other behaviours are simpler: if the drive for the sleep behaviour is the highest, the pig sleeps. If the drives for either fatigue or heat are the highest, the pig relaxes. If the fear drive is the highest, the pig moves in the opposite direction from the spotted lion: if the lion is to the west, the pig moves east, and so forth, including diagonal directions.

4.4 Optimising Drives and Initial Results

All results in this section were averaged over 6 runs of the simulation, on a terrain with 20 pigs and no people, and the default values for all other parameters as given in section 3. In all bar graphs, the T-shaped intervals represent the mean value plus or minus one standard deviation.

4.4.1 Probability of Changing Direction

In order to maximize the pigs' life expectancy, the `pig_direction_change` parameter was set to 0, after observing the results in Figure 3. This means that the optimal behaviour for a pig is to never change the direction it moves in when performing a random move: a random move is then just a movement in the same direction as the last, initially selected at random. This can be understood since a pig does not explore very much by running around in circles, which may happen if the direction it goes in keeps changing.

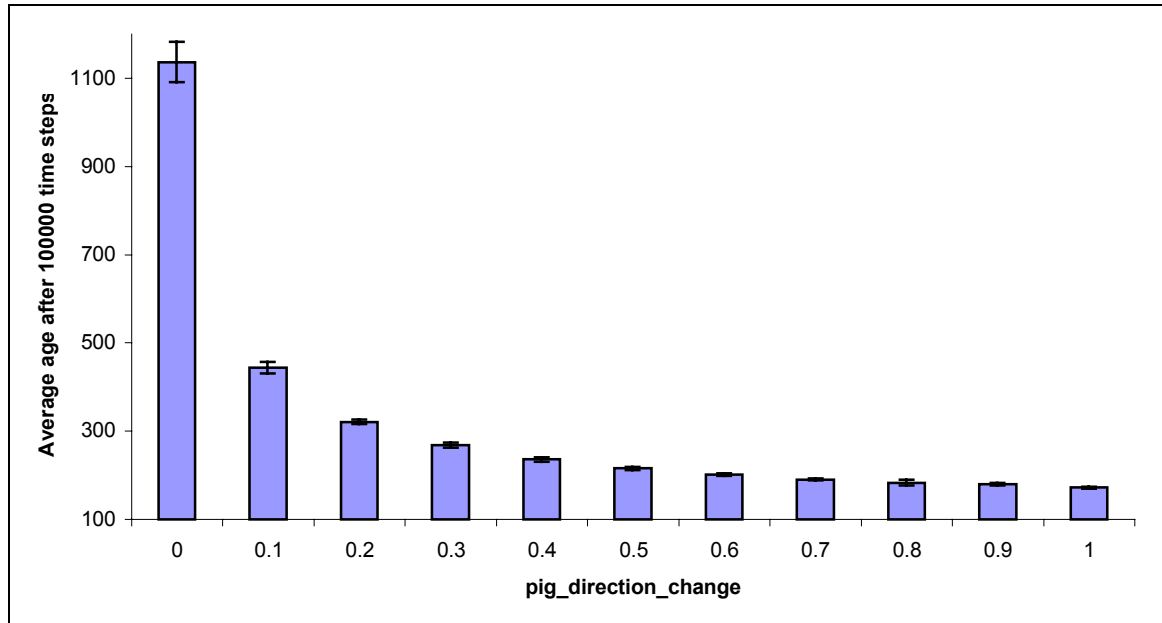


Figure 3: Impact of the **pig_direction_change** parameter on the pigs' average age after 100000 time steps, averaged over 6 runs.

4.4.2 Impact of Viewed Entities

According to the results in Figure 4, the **pig_impact_of_viewed_entities** (introduced in section 4.2) parameter was set to 0.9. This implies that whether a resource is visible or not is very relevant when choosing which behaviour to perform.

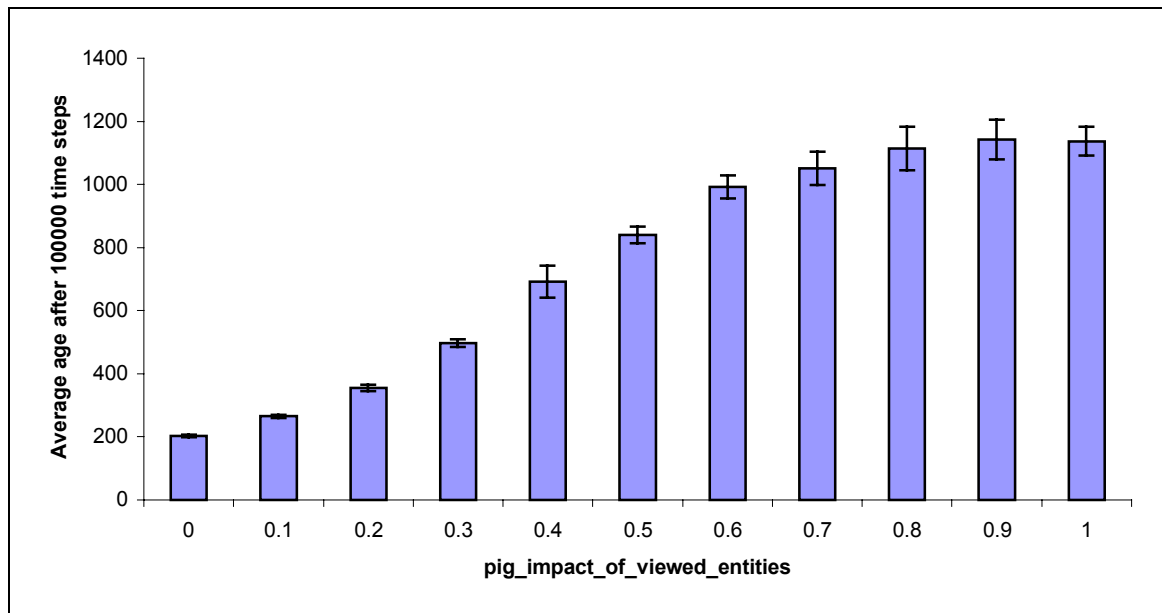


Figure 4: Impact of the **pig_impact_of_viewed_entities** parameter on the pigs' average age after 100000 time steps, averaged over 6 runs

4.4.3 Optimal Drives Results

Figure 5 shows the average and the moving average age achieved by pigs. The moving average over 100 pigs is the average age of the last 100 pigs that died; the average is over all pigs since time step 0.

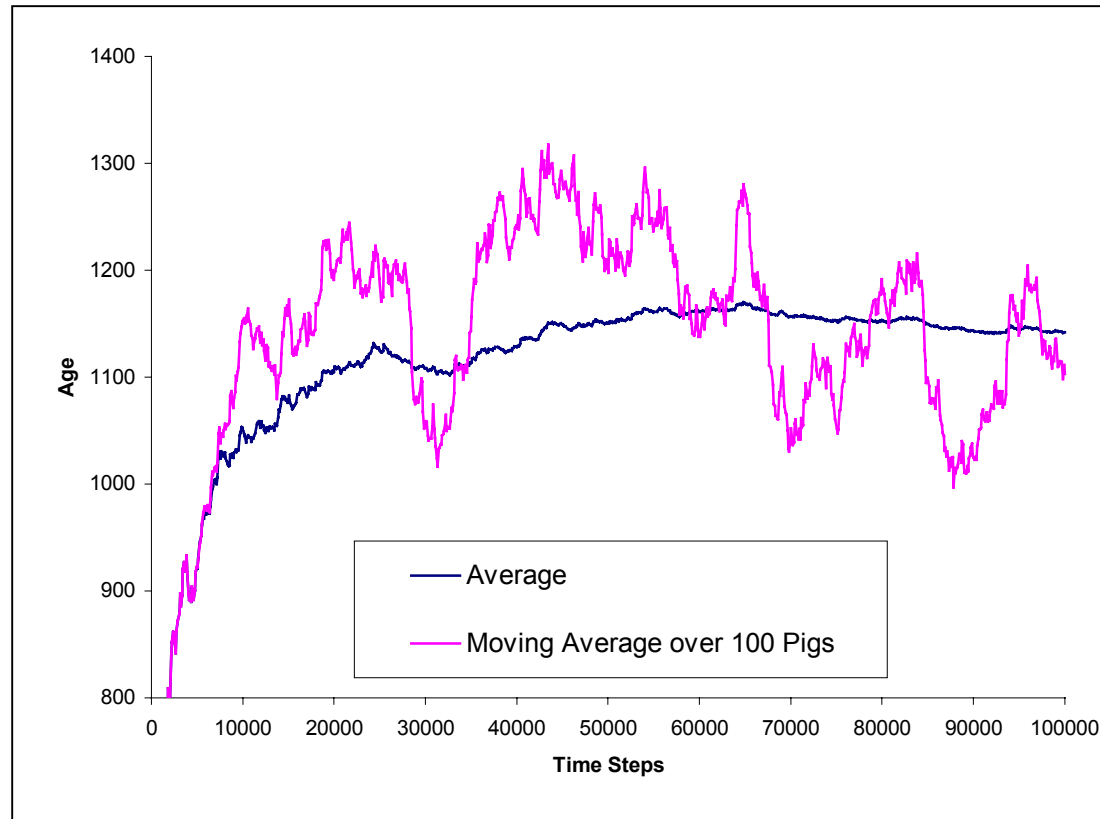


Figure 5: The average and moving average age achieved by pigs after 100000 time steps, with no people in the environment, averaged over 6 runs. The maximum standard deviation from the average age at any time step over the 6 runs is 273 time steps of age.

The average age achieved by the pigs at the end of the simulation is about 1142 time steps, with a standard deviation of 65 time steps. Note that both the average and the moving average age increase considerably at the start of the simulation. This is due only to the fact that they are initially set to 0, and the first pigs to die tend to have had a shorter life span than the average.

One likely cause for the oscillations in the moving average is the fact the clusters of slop and water move around the terrain. It is advantageous for the pigs when large clusters of slop and water are close together, as the pigs spend more time consuming and less time wandering, as the journey is short between the two clusters. However, the clusters will vary more quickly when this happens, as they are consumed more rapidly. Another possible cause for the oscillations is that the *availability* of the resources needed by the pigs varies with time. What is meant by availability in this case is that the more evenly spread a resource is in the terrain (not densely clustered), the more chance a pig has of encountering it when needed, and

that resource is thus more available. The noise observed in both the average and moving average is to be expected, given the randomness in the simulation.

4.4.4 Cause of Death

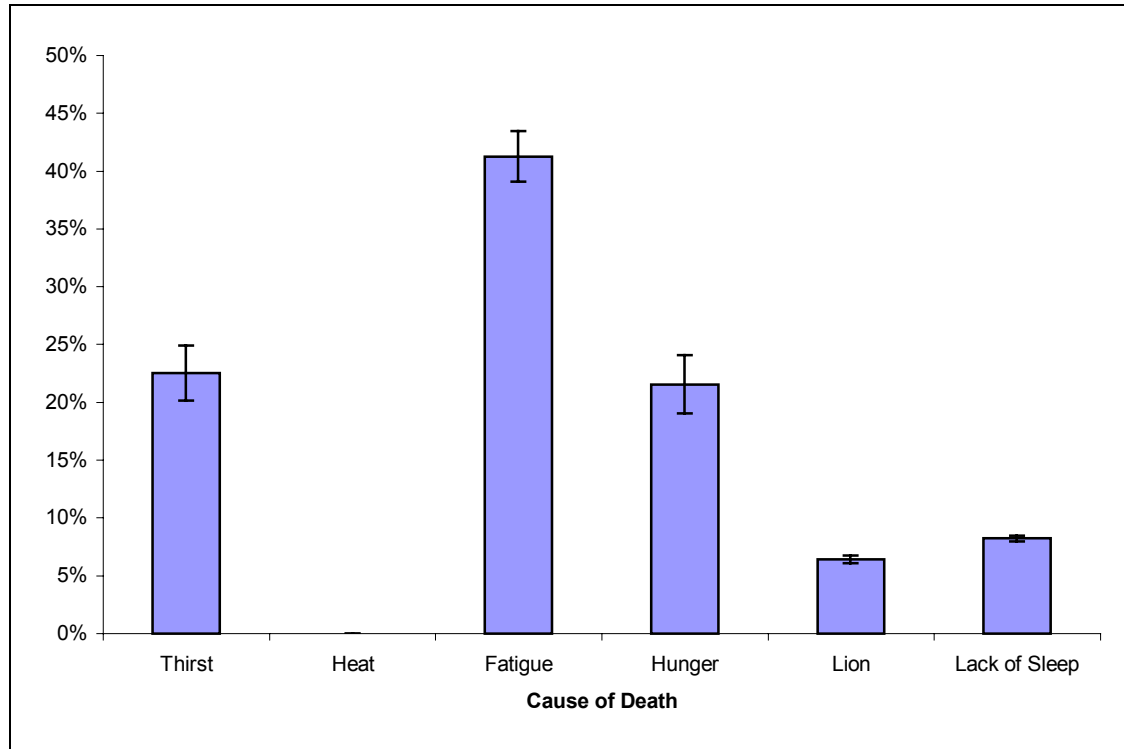


Figure 6: Causes of death in pigs after 100000 time steps.

Figure 6 shows what proportion of pigs dies of each possible cause of death. Most died of fatigue, which is to be expected when a pig wanders about without being able to find resources to satisfy its needs. The same amount of pigs died of hunger and thirst, which is normal as both resources are required in the same quantities, and are equally available in the terrain. The lion managed to catch a pig roughly every 500 time steps, but other deaths may also be associated with the lion: if a pig is running away from a lion, it is not able to pay any attention to its other stimuli. No pigs died of heat because they would die sooner of fatigue or lack of sleep: the internal sleep and fatigue levels increase faster than the heat level, and both sleeping and resting reduce the heat level (see section 3.4.2).

5 W-Learning in People

Before describing the W-learning action selection mechanism used for the people in ‘pigs and people’, we must introduce reinforcement learning, and more specifically a particular implementation of reinforcement learning: Q-learning. W-learning builds upon Q-learning to create a complete, hierarchical action selection mechanism.

5.1 Learning

What is learning? Generally, learning can be defined as the acquisition of skills by experience or study. In our case, we will use the more specific definition given by Mitchell [1997]. This definition applies to learning within a computer program: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E ”. We will first introduce reinforcement learning, and then examine Q-learning, which is a particular form of reinforcement learning. Finally, we shall explain W-learning, which uses Q-learning to build a complete action selection mechanism. In section 6, we will examine whether W-learning is an effective learning mechanism.

5.1.1 Reinforcement Learning

The workings of reinforcement learning are intuitively comparable to how one could imagine learning, say, in any young animal learning to fend for itself. This animal may burn itself and learn to avoid hot things in future. In machine reinforcement learning, a numerical reward is sensed instead of the burning feeling. The reinforcement may be a reward, nothing, or a penalty (as above) that may teach the agent in question to avoid moving towards a sensed heat source in future.

Sutton and Barto [1998] give a complete introduction to the field of reinforcement learning. They define reinforcement learning, for an agent, as “learning what to do – how to map situations to actions – so as to maximize a numerical reward signal.” In reinforcement learning the agent learns what to do by interacting with its environment. The agent is not told what to do, and does not find out whether the action it chose to perform was the best of all possible actions. The agent simply receives reinforcement after performing its chosen action. Based on the reinforcement received by performing different actions in different states, the agent constructs a policy, which is essentially a mapping from situations to actions. Policies are iteratively evaluated and optimised in order to converge towards the optimal policy for the environment. The optimal policy is the policy that provides the maximum obtainable reward.

A problem that arises throughout reinforcement learning is commonly known as the exploration vs. exploitation problem. The issue is to balance the use of existing knowledge with the acquisition of new knowledge: in any given situation, should the

agent make the best possible move according to its current policy, or try another action which may lead to some more rewarding state and thus become the best action to take in that situation? Three solutions to this problem are known as ϵ -greedy, greedy with optimistic starts, and softmax.

ϵ -greedy works by selecting the best possible action with probability of $1 - \epsilon$, and choosing another action randomly with probability ϵ , where ϵ is some small positive number less than 1. ϵ can be either constant, or can decay with time making exploration more frequent earlier on, and exploitation more common later when the agent has more experience.

Greedy with optimistic starts works by setting the initial values in the state-action table to higher than possible values. Thus, the first time each action is attempted at every state, its value will inevitably decrease, and all the untried actions will then have higher values. This allows each action to be evaluated at least once, which is necessary to ensure convergence towards the optimal policy.

Softmax selects actions with a probability dependent on their estimated value; for example, we could use a Boltzmann distribution to achieve this:

$$\frac{e^{Q(a)/\tau}}{\sum_{b=1}^n e^{Q(b)/\tau}}$$

Here, $Q(a)$ is the estimated value of taking action a at the current time step, n is the number of possible actions, and τ is known as the ‘temperature’. τ is a positive, adjustable parameter:

- A high value of τ makes all actions nearly equally probable,
- Low values of τ cause the action with highest estimated value to have a much higher chance of being selected than the others.

5.1.2 Q-learning

Q-learning implements a reinforcement learning solution by building lookup tables of state-action pairs. For each of these, a Quality-value $Q(s, a)$ is learnt, which we shall term Q values from now on. For any given state, the action with the highest associated Q value is the estimated best action to perform. Any of the mechanisms described in section 5.1.1 can be used to balance the exploration vs. exploitation problem. After an action, selected at time t , is performed, a reward is received at time $t+1$. The cell in the state-action table corresponding to the state at time t and the action chosen is then updated following the formula:

$$Q(s_t, a_t) = (1 - \alpha) \times Q(s_t, a_t) + \alpha \times (r_{t+1} + \gamma \times \max_a Q(s_{t+1}, a))$$

- r_{t+1} is the numerical reward received at time $t+1$.
- α , the **learning rate** is a parameter that can be set to vary the effect that new experiences have on existing knowledge. The higher the α , the more impact recent experiences will have on the corresponding Q values. α must decay with time in order for the policy to converge. The learning rate for Q values is from now on called the Q learning rate.
- γ , the **discount factor** determines the importance of future rewards: the higher the γ , the further the agent looks ahead at any point in time to estimate the reward it will receive.

However, in any simulated environment with an interesting amount of complexity, the state action space will be far too large to store explicitly in a lookup table. Some form of generalization is required. Neural nets are an obvious solution to this problem, and are what Humphrys used in his implementation when a full state-action space was used, as the size of the full state-action space was 10.8 million in his house robot problem [Humphrys, 96a].

As Humphrys notes, reinforcement learning has most often concentrated on problems with a single goal; however, any real action selection problem often has many heterogeneous and possibly conflicting goals all competing in parallel. Indeed, in his first experiment, [Humphrys, 96a] shows that using a single monolithic Q-learning agent is not very effective as an action selection mechanism. However, when a number of sub-agents are used, each competing to have its action executed, performance is very satisfactory. Here, each sub-agent is like a behaviour as introduced in section 4.1. Long-Ji Lin introduced “hierarchical Q-learning” [Lin, 93], where each sub-agent proposes an action, and a *learning switch* decides which sub-agent gets to perform its chosen action. This learning switch needs a global reward function to learn which agent to select at any given time. It is easy to see that Tyrrell’s affirmation that hierarchy was desirable in an action selection mechanism is supported by this case, and indicates that it may be just as valid for learning mechanisms as non-learning mechanisms. The theory behind hierarchical Q-learning is that each sub-agent faces only a portion of the overall problem, and only needs to learn what is relevant to its behaviour.

5.1.3 W-learning

W-learning builds on Q-learning to produce a complete action selection mechanism. In hierarchical Q-learning, the sub-agents are organised by a global reward, the sub-agent whose action is selected being the one that is expected to give the highest contribution to the global reward. Humphrys [1996b] notes that the sub-agents do not need a global reward, and that they can perform similarly using a competition based on using a weighted value of local rewards. He therefore proposes W-learning. In W-learning, the local rewards are specific to each sub-agent; and are produced by

individually hand-coded reward functions based on each sub-agent's goals. By attempting to satisfy all the sub-agents' goals, the agent will ideally achieve its overall goal.

W-learning eliminates the need for a learning switch, introduced in section 5.1.2. Instead, each sub-agent proposes an action with an associated weight: a W value. W values are learnt at the same time as the Q values, although there is only one W value for each state. W values represent the estimated loss that a sub-agent incurs when its chosen action is *not* performed. They are initialised to 0 or some small random negative numbers. W values are updated at each time step, for each agent i that did not have its selected action performed, following:

$$W_i(x) = (1 - \alpha_w) \times W_i(x) + \alpha_w \times (1 - \alpha_Q)^\omega \times (Q_i(x, a_i) - (r_i + \gamma \times \max_{b \in A} Q_i(y, b)))$$

- $W_i(x)$ is the W value for agent i at state x .
- α_w is the learning rate for W values.
- α_Q is the learning rate for Q values.
- ω , the **delaying rate** is particular to W learning, and is an attempt to specify how well the Q values should be learnt before learning the W values.
- r_i is the reward obtained at the current time step by agent i . There is no need for negative reinforcement; Humphrys [1996b] shows that the all behaviours possible within W-learning can be learnt using only positive rewards.
- The rest of the variables are the same as those described in the previous section on Q learning.

The concept of the delaying rate is introduced fully in [Humphrys, 95]. Suffice to note that a low delaying rate allows the W values to be learnt quickly, and a high delaying rate delays learning of the W values until the Q values are well known.

Various methods for choosing actions are possible within W-learning. The simplest is choosing the action that would suffer the most if not selected. This is known as 'Minimize the Worst Unhappiness'. Other selection methods include:

- 'Maximize the Best Happiness', in which case the sub-agents promote their action in proportion to the associated Q values. This is equivalent to selecting agents based on their current Q values alone, ie setting $W = Q$.
- 'Maximize Collective Happiness' and 'Minimize Collective Unhappiness', with the obvious meanings, with 'Collective' referring to all the sub-agents.

- ‘Negotiated W-learning’ where sub-agents have several rounds of negotiation before producing a winning action. Nothing is permanently learned, everything being evaluated from scratch in each negotiation round. This performed the best of all experiments in Humphrys’ paper.

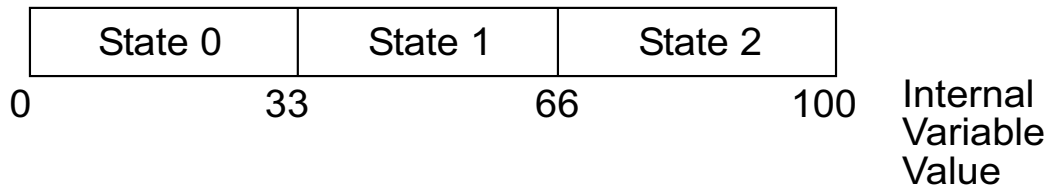
Negotiated W-learning is of little interest in our case, as the sub-agents must share the same action space. In the house robot environment, the robot had to select only which direction to move in. In pigs and people, the people have a wider repertoire of actions: nine move actions, plus eat, sleep, drink and relax.

5.2 The Implementation of W-learning

Having introduced W-learning, we will now detail the implementation used in people. One sub-agent was used for each type of behaviour a person needed to exhibit. Each sub-agent senses a different state-space, and receives feedback from a distinct reward function. The ϵ -greedy method was used to solve the exploration vs. exploitation.

5.2.1 A Person’s State-Action Spaces

A parameter (internalVariableStateGranularity) was used to determine to what level of detail the W-learning mechanism senses the people’s internal stimuli. Unless otherwise specified, this parameter was set to two, effectively splitting the internal stimuli space into three states:



State-action lookup tables were introduced in section 5.1.2. Given three states per internal stimuli, the heat, sleep and fatigue sub-agents have a state-action table like:

State \ Action	0	1	...	9
0	Q(0, 0)	Q(0, 1)	...	Q(0, 9)
1	Q(1, 0)	Q(1, 1)	...	Q(1, 9)
2	Q(2, 0)	Q(2, 1)	...	Q(2, 9)

The state-action tables for drinking and eating are similar, except they also encode the direction to the nearest food. This can take on 10 possible values: eight possible directions (includes diagonals), currently on resource, and no resource visible. Thus, the tables for eating and drinking are 10 times larger than the other

tables: for each state of the internal stimuli for hunger, there are 10 entries, one for each direction. The fear table however takes no account of the internal fear level, and has 10 state entries, which are for the direction of the nearest lion.

We thus have $3 * 30 + 2 * 300 + 1 * 100 = 790$ Q values to learn, because we are using sub-spaces of the total state-action space. A complete state-action space would have had $10 * 3 * 3 * 3 * 30 * 30 * 10 = 810000$ Q values. Should the internal level of granularity, as set by the `internalVariableStateGranularity` parameter, be set to 3 this figure becomes 10240000, vs. only 1020 for W-learning with sub-spaces. The size of the state-action space for the full space is $(\text{granularity} + 1) ^ 5 * 1000$, which grows exponentially. However, using sub-spaces, we can reduce that to a formula that grows linearly: $230 * (\text{granularity} + 1) + 100$.

An example of a W table and how it encodes the state action space is given in appendix B, at two different stages in the learning process.

5.2.2 A Person's Reward functions

A distinct reward function is associated with each sub-agent as introduced in section 5.1.3. Each reward function returns a reward between 0 and 1. The functions for hunger, thirst, sleep, fatigue and heat give a reward only if the associated internal variable decreased after the last action was performed. The amount rewarded is directly proportional to how high the internal variable was in the previous time step, and values are directly related to the different states as described in section 5.2.1. For example, for thirst with an internal variable granularity of 2, the thirst reward function returns:

- 0 if the thirst level has not decreased (or has decreased by less than 20 in the fatigue and sleep reward functions, and by less than 10 in the heat reward function. This is due to the effect of actions on internal variables as described in section 3.4.2),
- 0.33 if the thirst level has decreased, and the thirst level was below 33 before the action was performed,
- 0.66 if the thirst level was between 33 and 66,
- 1 if the thirst level was between 66 and 100.

The fear reward function is the simplest of all. It returns 1 if a lion could be seen before the action was performed, and no lion can be seen after that action, in the current state. 0 is returned in all other situations.

Furthermore, the above rewards are scaled by factors between 0 and 1, as the W values that are learnt depend on the Q values in the state-action tables. Thus, a sub-agent receiving a higher scaled reward has more importance, and thus its actions will be performed more often. A genetic algorithm was used in section 5.3.5 to optimise the set of scaling factors, as these are parameters that need to be set by hand for the W-learning algorithm.

5.2.3 Passing Knowledge Down Generations

When a person dies, a new person is created in the environment. The initial population was given randomly initiated W-learning tables. (We will call the collection of W tables a person uses to select his actions ‘brains’ from now on.) However, for what people have already learnt to be useful to the new people appearing in the environment, acquired knowledge must be passed on in some fashion. Random initiation is not good enough, as learning is a slow process: people would perform very poorly if every one had to learn everything from scratch. The initial implementation kept a record of the 10 best brains; ‘best’ being judged by how long the person using that brain had managed to survive in the environment. Each new person was given a random selection of one of the best brains, which he could improve by learning. This resembled somewhat a genetic algorithm approach, where the elite individuals were never culled. This was deemed rather unrealistic, and furthermore, reduced the possibility for individuals to adapt should their environment change. Thus, a new mechanism was devised.

When a new person is created in the current implementation, he receives a brain from a random selection of one of the brains currently being used by the people alive in the environment. This means that a new person will receive a brain recently updated and currently used by another living individual, likely to be appropriate for the current state of the environment. The longer a person lives, the more of a chance he has to propagate his brain to the next generation, and the more times he may do so. If a person has a brain unsuited to the environment, he will die off quickly. Thus, an unsuitable brain will have a very slim chance of being given to anybody. The environment therefore determines the fitness of a brain, which is a good thing as this is much more reliable than a fitness function coded by a designer. However, not all the information in a brain should be passed on, as the new person must forge his own experiences.

Which information should be passed on between generations, and how much forgotten? The Q values within the tables are passed as they stand, since the optimal action to perform for a given behaviour at a given state is unlikely to change. However, the learning rate is reset, which gives the individual receiving this information a chance to update it following his own experience, should it diverge from what is represented by the current state of the tables. The tables counting the number of times a state has been visited are reset because the new person has not experienced them. This has a similar effect to resetting the learning rate: the entries in these tables are used to update both the Q and W values, as the Q learning rate depends on the number of times a state has been visited as described in section 5.3. The W values are passed after being discounted by a factor of between 0 and 1; this is optimised in section 5.3.4. This discounting should encourage new people to adapt their choice of behaviour to the current state of the environment. To the same effect, ϵ is reset to restart the ϵ -greedy in the selection of actions from a table.

5.3 Optimising W-Learning and Initial Results

Unless specified otherwise, all results obtained in this section are averaged over 6 runs of the simulation with no pigs for 100000 time steps. The resource parameters described in section 3.2 were used, and there was one lion in the terrain. The Q learning rate for a given entry in the table is the inverse of the number of times that entry has been visited by the agent, plus 1 to avoid division by 0. Visiting an entry (s, a) in a state-action table in this case means performing action a in the state s , and updating $Q(s, a)$ according to the reward received. The variant of W-learning used was ‘minimize the worst unhappiness’, introduced in section 5.1.3.

The T-shape intervals at the top of bar graphs represent the average plus or minus one standard deviation over the different runs.

5.3.1 Discount Factor

The following graph shows that the discount factor, described in section 5.1.2 and used in the update formulae for both Q and W values, has little effect on performance, as long we keep it away from the extremes of 0 and 1 which assign no importance to future rewards and as much importance as the current reward, respectively. The discount factor was set to 0.6 in the rest of the experiments, although any value between 0.1 and 0.9 would have done (Figure 7). 0.6 was chosen as this was the value used in [Humphrys, 96a].

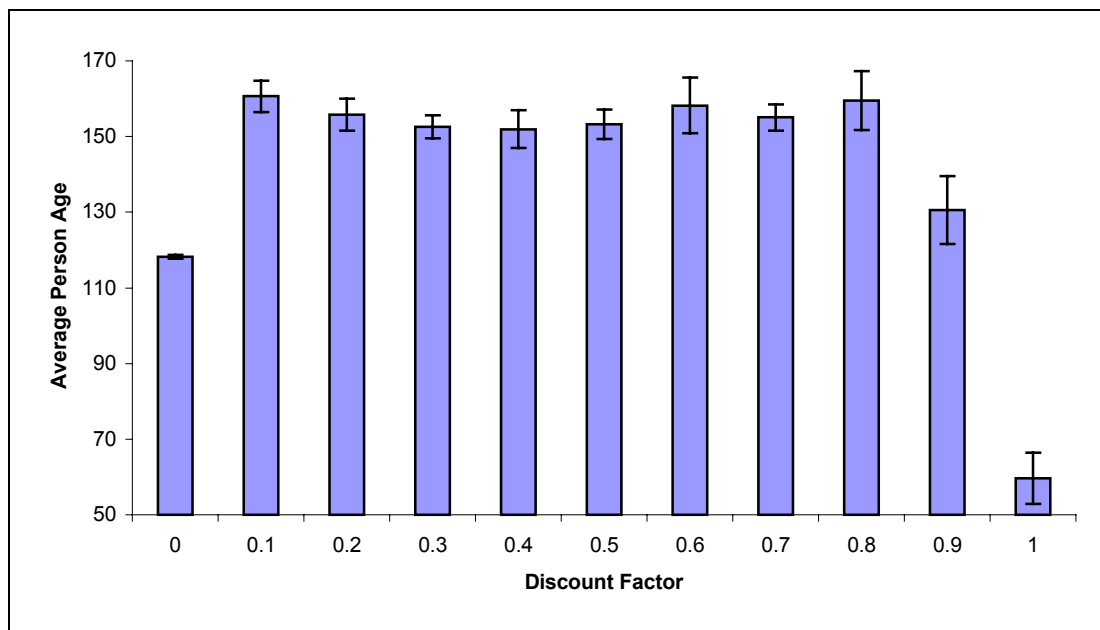


Figure 7: Variations in the discount factor averaged over six runs of 100000 time steps.

5.3.2 Delaying Rate

The delaying rate, introduced in section 5.1.3, has little impact on the people's performance as shown in Figure 8. The solution used throughout the rest of the experiments was to start learning the W values immediately (delaying rate set to 0).

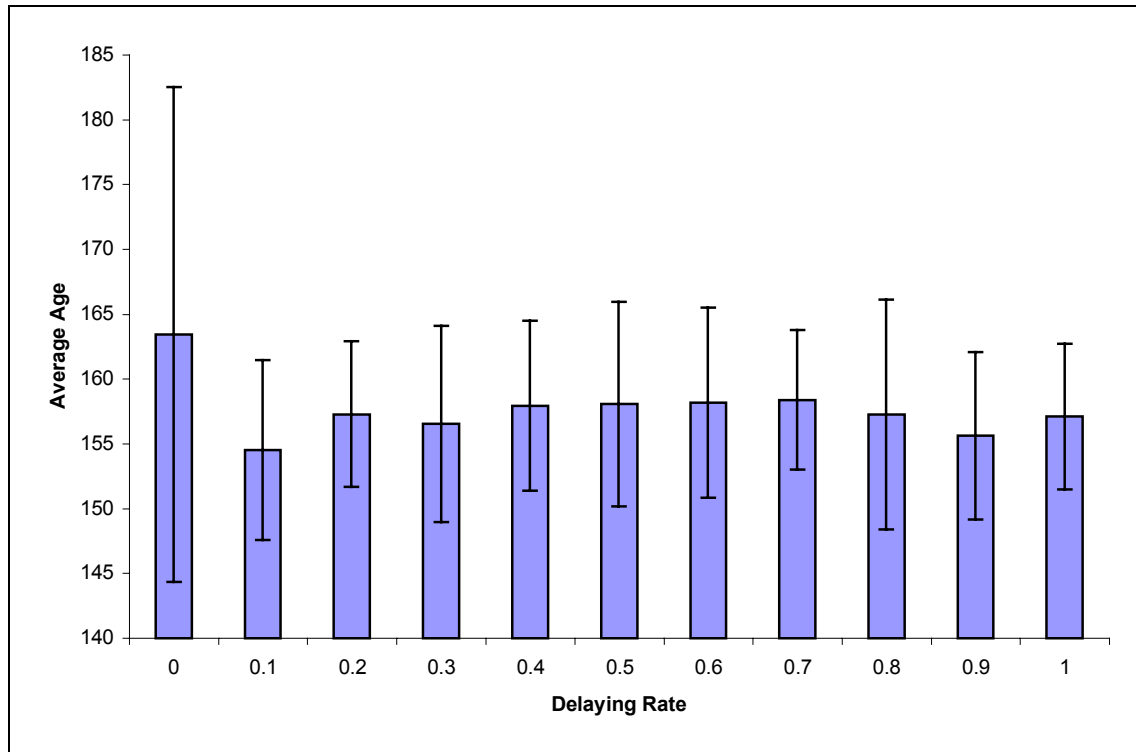


Figure 8: Variations in the delaying rate averaged over six runs of 100000 time steps.

5.3.3 AlphaW and Initial AlphaW

In these experiments α_w (alphaW), introduced in section 5.1.3, is initialised to 1 and is updated at each step following:

$$\alpha_{w_{t+1}} = \frac{1}{(\alpha_{w_t})^{-1} + 1}$$

Thus, for the first people created in the environment, α_w takes on values 1, 1/2, 1/3, 1/4 and so forth, mimicking [Humphrys. 95]. `initial_alphaW` is a parameter that determines what α_w is reset to when W tables are passed on to a new person as described in section 5.2.3.

In these, experiments, `initial_alphaW` was set to a fixed optimised value; however, it would be interesting to attempt to find a mechanism to set this dynamically, based on a measure of the experience of the person passing the brain. The results obtained when optimising `initial_alphaW` are in Figure 9.

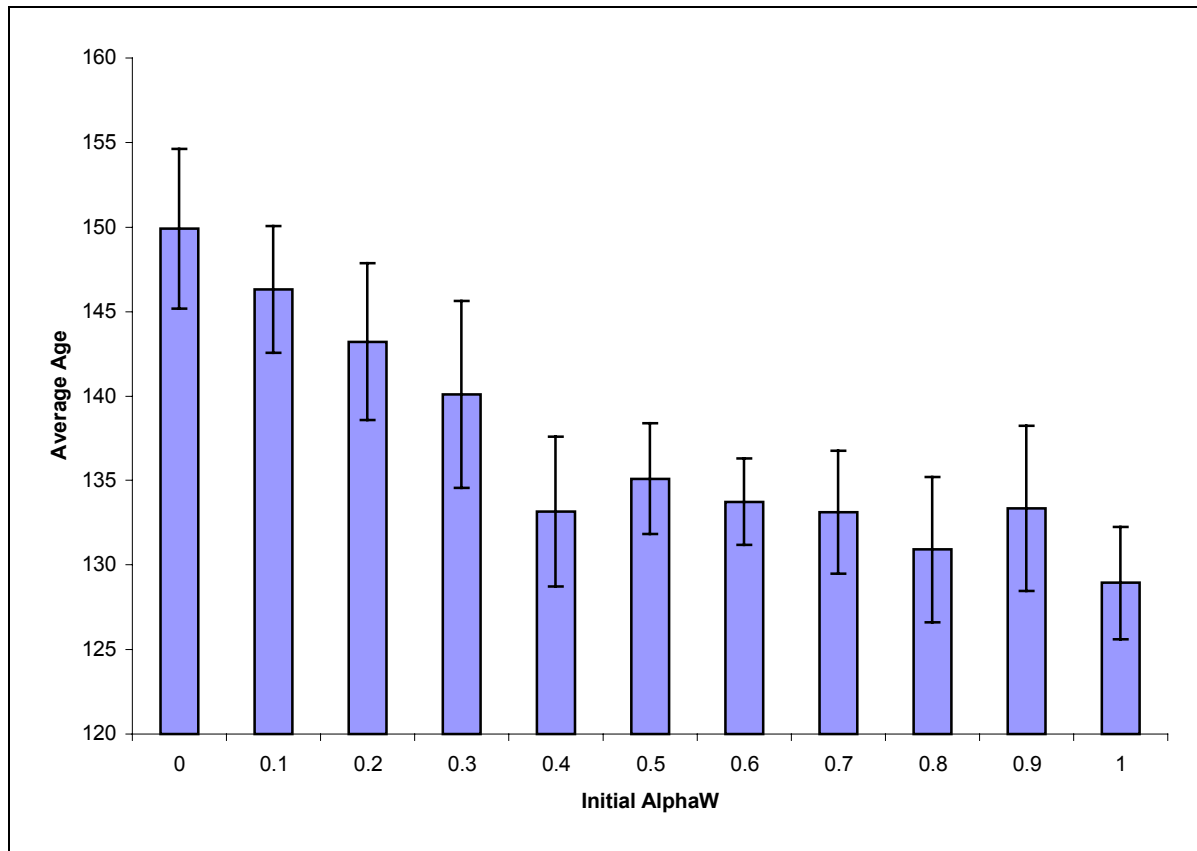


Figure 9: Variations in initial `alphaW` averaged over six runs of 100000 time steps.

The value used for `initial_alphaW` in the rest of this paper is in fact 0.03, which was found to perform slightly better than 0 on average, but there is no true statistical difference.

5.3.4 W Value Discount Factor

Figure 10 is a graph of the optimisation performed for the discount factor applied to the `W` values from one generation to the next. Any value would do, so the `W` value discount factor was arbitrarily set to 0.6 from here on.

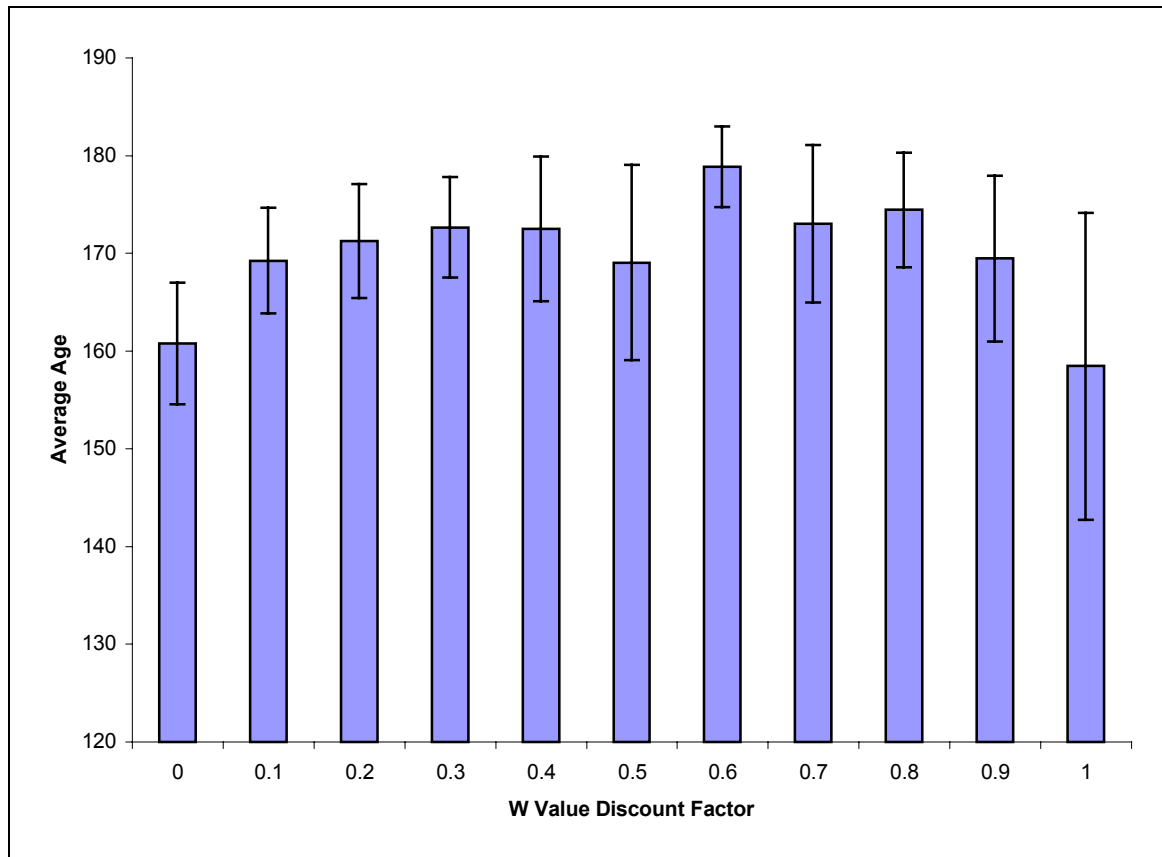


Figure 10: Variations in the W value discount factor averaged over six runs of 100000 time steps.

5.3.5 Reward Function Scaling Factors

The optimal values for the reward scaling factors were searched for using a genetic algorithm. All the values given by the optimisations in sections 5.3.1 to 5.3.4 were used. The best factors obtained are:

- Thirst = 0.93,
- Hunger = 0.11,
- Fear = 0.46,
- Sleep = 0.83,
- Heat = 0.01,
- Fatigue = 0.45.

This **genetic algorithm** optimisation took place on an environment with 20 people and 20 pigs, as the presence of pigs may affect the importance of the behaviours. A population of 10 chromosomes was optimised for 30 generations, the fitness for each chromosome being the average age achieved by the people. Each chromosome had six genes, one for each scaling factor. Each gene was represented by 7 bits; this allowed for a possible 128 evenly distributed distinct values between 0 and 1, inclusive, for each scaling factor. The fitness for each chromosome was originally set to be the average age after 100000 time steps averaged over six runs of the simulation instead of just one run; but each generation took an exceedingly long amount of time to evaluate. The numbers given above may thus not be optimal.

5.3.6 Optimal W-Learning Results

Figure 11 is a plot of the average and moving average of the people's age, after all the optimisations performed throughout section 5.3, with no pigs in the environment.

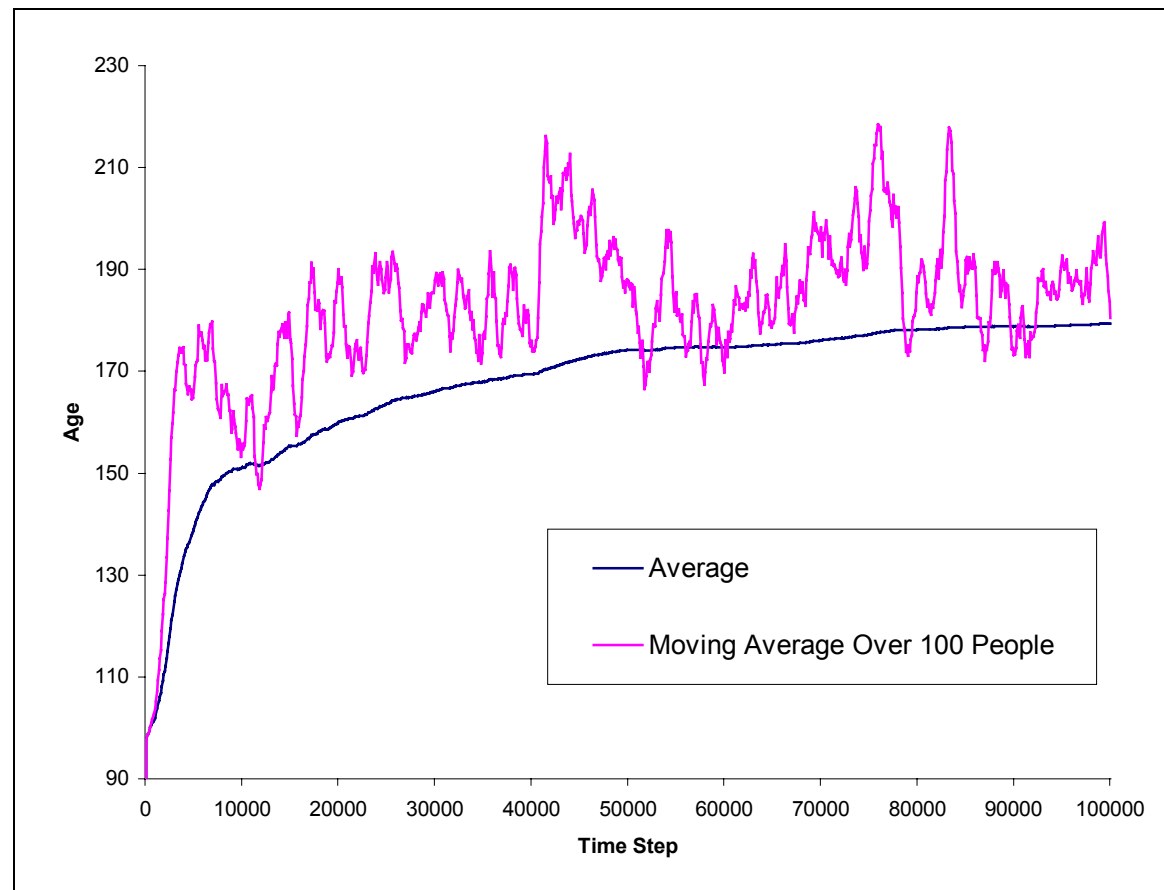


Figure 11: The average and moving average age achieved by people after 100000 time steps, averaged over 6 runs. The maximum standard deviation from the average age at any time step over the 6 runs is about 19 time steps of age.

People achieve about 179 time steps as their final average, after 100000 time steps. The standard deviation at this final time step was 6.5 time steps. Looking at the moving average, the sharp increase in the first few thousand time steps is likely to be when most of the learning is taking place. Throughout this short period, many generations die, as the adequate behaviour has not yet been learnt. All experience is also new, and thus the behaviour changes more rapidly, becoming progressively more adequate for the environment. Learning still takes place during the rest of the simulation, but has less effect than at the beginning, and can be considered as more of an optimisation process of the existing behaviours.

5.3.7 Causes of Death

Figure 12 shows that people died almost exclusively of thirst and hunger.

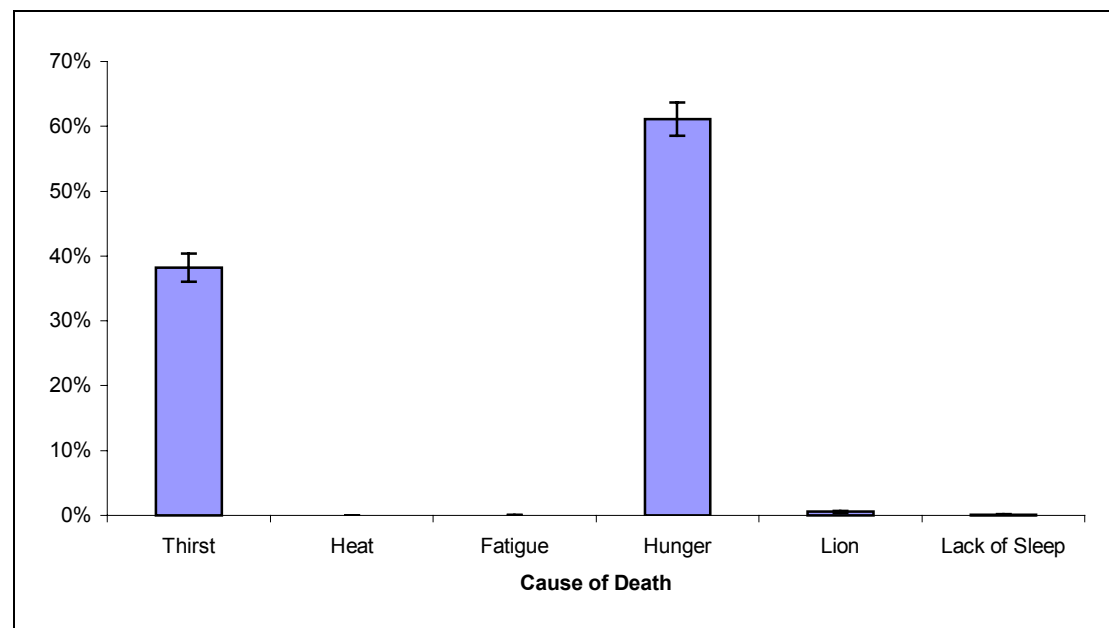


Figure 12: The causes of death in people after 100000 time steps.

More people died of hunger than thirst, this can be understood by observing the scaling factors for the reward functions optimised in section 5.3.5. Indeed, people are rewarded more for drinking, and thus give this behaviour more priority. The fact that hunger and thirst account for nearly all of the people's deaths implies that people successfully learn to maintain their other internal variables to appropriate levels, as the actions necessary to do so can be performed at any time and in any situation in the terrain. However, the people do not seem to have learnt to find and consume wheat and water very well.

6 Experiments

Unless specified otherwise, the following experiments were performed with 20 pigs and 20 people, and 1 lion in the terrain. Each result is average over 6 runs of 100000 time steps. The resource parameters described in section 3.2 were used:

- The number of entities of each resource was set to 90,
- The proximity_probability parameter for each resource was set to 0.6.

The drives mechanism was used with the parameters optimised in section 4.4:

- Probability of direction change = 0,
- Impact of viewed entities = 0.9.

The variant of W-learning used was ‘minimize the worst unhappiness’ with the parameters found in section 5.3:

- Discount factor = 0.6,
- Delaying rate = 0,
- Initial $\alpha W = 0.03$,
- W value discount factor = 0.6,
- Reward function scaling factors as given in section 5.3.5.

Statistics evaluated include the average and the moving average ages of the species, and the proportion of animals that died of each possible cause of death. Here, the moving average for a species is the average age of the last hundred individuals of that species at their time of death. The moving average gives information about how a species is performing at different times within a run. The average only provides information on performance over a whole run, and thus always includes the initial individuals who died off at a young age early on in the simulation.

6.1 Pigs vs. People Apart

As the action selection problem is the same for pigs and people, the results obtained in sections 4.4.3 and 5.3.6 can serve to compare the two mechanisms. Figure 13 combines the results obtained in these sections into one graph.

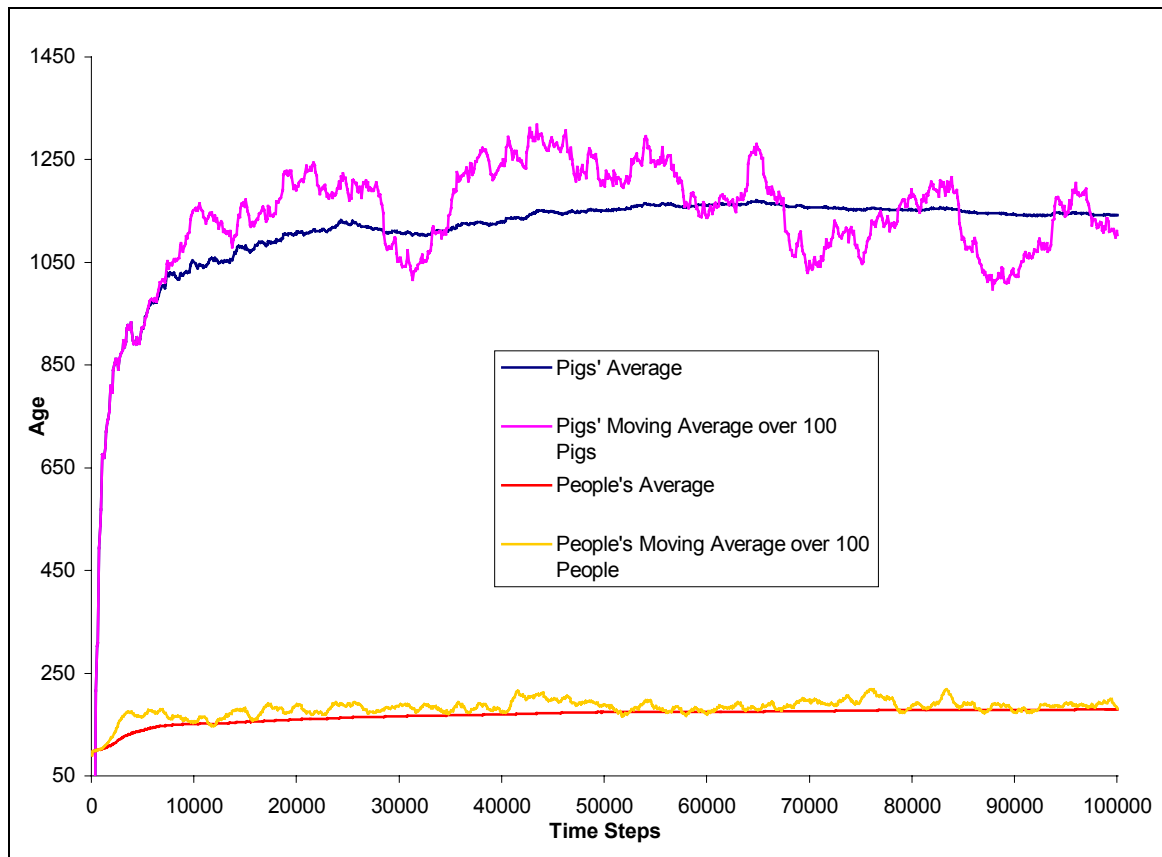


Figure 13: Pigs and people's performance when the other species is not in the environment.

The pigs perform far better than the people: their average age at the end of the simulation is about 1142 time steps, they live over 6 times longer than the people who achieve only about 179 time steps as their final average. For the average ages, the standard deviation from the mean was 63 time steps for the pigs and is due only to the randomness in the terrain. The standard deviation was 6.5 time steps for the people. The difference in performance between pigs and people is thus highly significant. The people's performance in all experiments is however nearly twice that achieved by an agent making random moves (not figured in above graph) which achieved a final average age of 93.7 with a standard deviation of 0.07.

Why is the pigs' performance so much greater than that of the people? Can the people be made to perform better in any way? We will try to answer these questions throughout the rest of this section.

6.2 Pigs vs. People Together

It is more interesting to compare the mechanisms with both pigs and people in the environment. This ensures that both types of animals are facing the same problems on average throughout the simulation.

6.2.1 Results

Figure 14 gives the results obtained with both species coexisting in the environment. As before, the pigs seriously outperform the people. At the last time step of the simulation, the pigs' average age is 1294 time steps with a standard deviation of 84 time steps, and the people's is 171 time steps with a standard deviation of around 4.7.



Figure 14: Pigs and people's performance when the other species is present in the environment.

The first thing to notice is that the pigs' performance improves when there are people in the terrain, but the people's performance decreases. The student t-test was performed on final average ages at the last time step, for both pigs and people, to compare the runs in this section and section 6.1. For people, the t-test gave $p < 0.0135$ which implies that the two series have not been drawn from populations with equal means with over 98% certainty. For pigs the t-test gave $p < 0.0024$ which also implies that the populations were indeed different. So there is a significant difference. Why does the pigs' performance increase and the people's decrease? Examining the causes of death in pigs and people may help explain this.

6.2.2 Cause of death

Figure 15 shows what proportions of pigs and people died of, both in the case with only one species in the environment (labelled 6.1), and with both species coexisting (labelled 6.2).

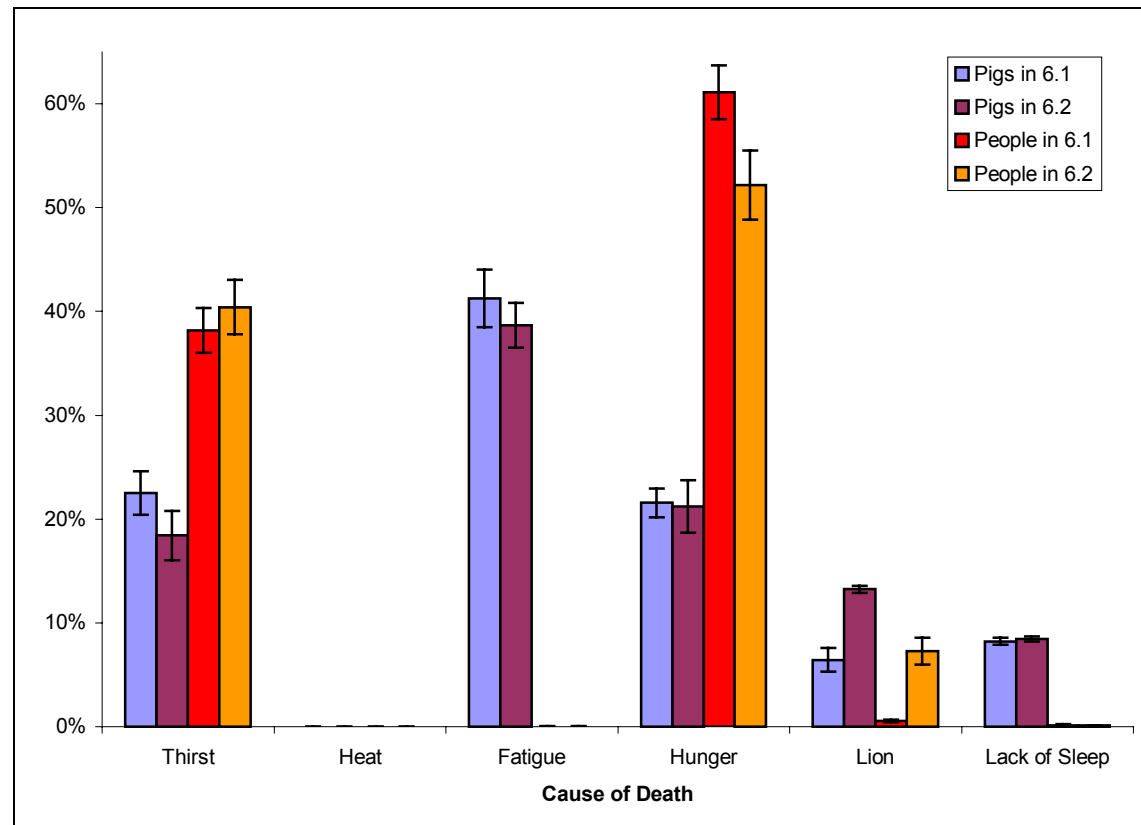


Figure 15: The causes of death in both pigs and people coexisting in the environment, after 100000 time steps.

There are no obvious clues that may help to explain the differences between the two runs expressed in section 6.2.1. Although both animals consume the common resource water, water is always available in the same quantity. However, the availability may vary, as described in section 4.4.3. The fact that the availability of water may change more quickly with two species consuming the resource could be more favourable to the pigs, but there is no true evidence to support this.

Being eaten by the lion is a cause of death for which there is a significant increase between the two experiments for both species. This requires further investigation, as the opposite would be expected: as the lion chases pigs and people indiscriminately, the proportion of time it spends chasing each species should be reduced. Thus, intuitively, a smaller proportion of animals of each species should be caught. Further investigation was not possible to date, due to time constraints.

6.3 A Dry Terrain

Figure 16 shows how both pigs and people performed in an environment with 5 times less water than in previous experiments.

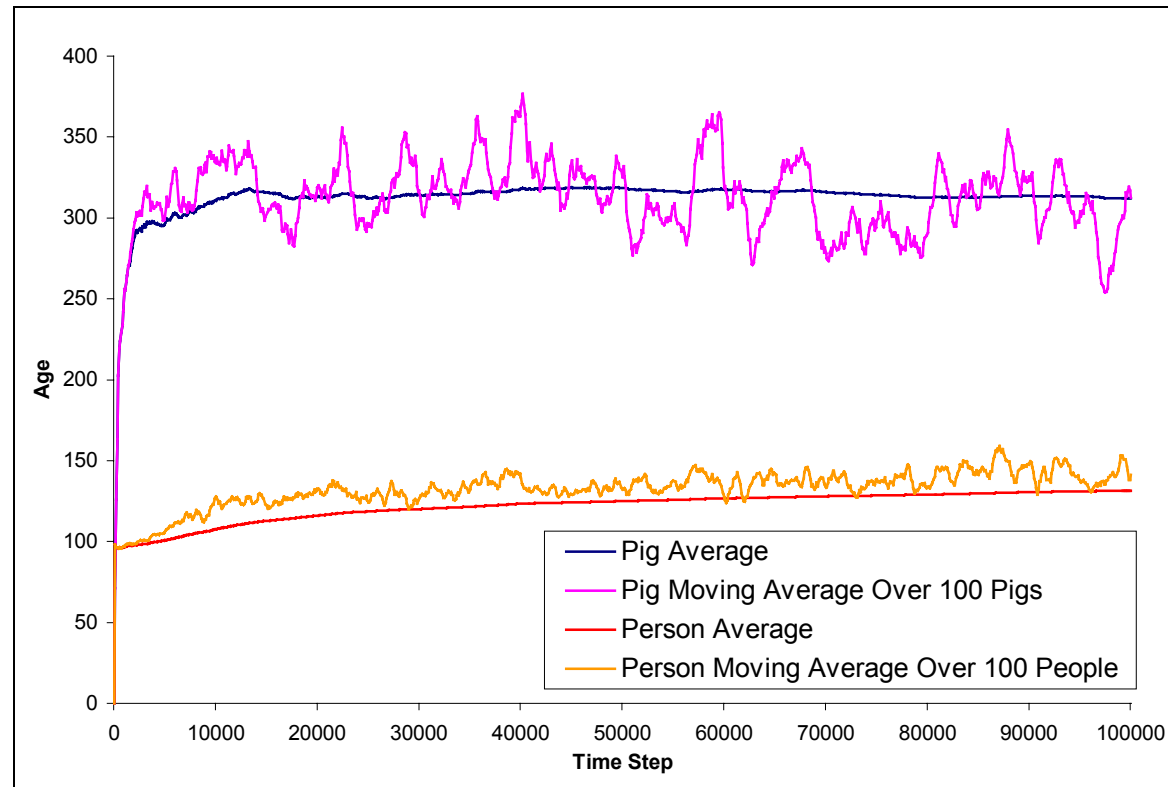


Figure 16: Performance of pigs and people in an environment with only 18 water entities.

Pigs show a decrease in performance of 76%, over three times as much as the people's 23%. Although more heavily affected, pigs continue to steadily outperform people. For pigs, who rely on the presence of water to achieve their high performance, the shortage of water is a serious problem. This is less the case for people, who were not very good at finding and consuming water in the first place. The fact that pigs take a bigger performance hit than people can therefore be explained by the large proportion of people that die of thirst, even when there is plenty water available, as in section 6.2.2.

As would be expected, Figure 17 shows that the proportion of animals that die of thirst increases in an environment with little water available. Both species appear to be equally affected. The decreases in the percentage of animals dying of all other possible causes of death are due to the fact that the animals die of thirst first.

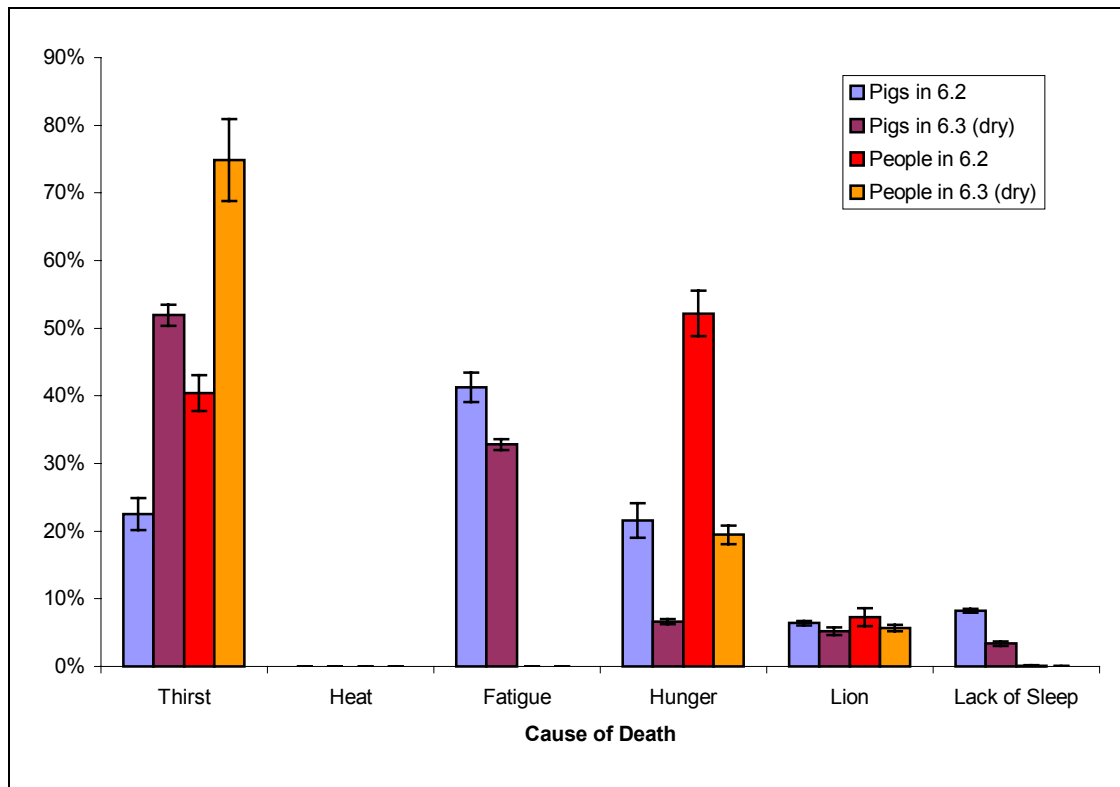


Figure 17: Causes of death in pigs and people in a terrain with only 18 water entities.

6.4 Varying the internal state-action space

So why do the people not perform better? One idea is that the W-learning mechanism perceives a more restricted state-space than the drives mechanism. How accurately the state is perceived by each sub-agent, except the one rewarded for moving so that the lion is no longer in sight, depends on the internalVariableStateGranularity parameter, described in section 5.2.1.

Figure 18 is a plot of the people's performance with different values for the internal stimuli granularity parameter, varying from 0 to 5, in the same environment as used in section 6.2 with 20 pigs and 20 people.

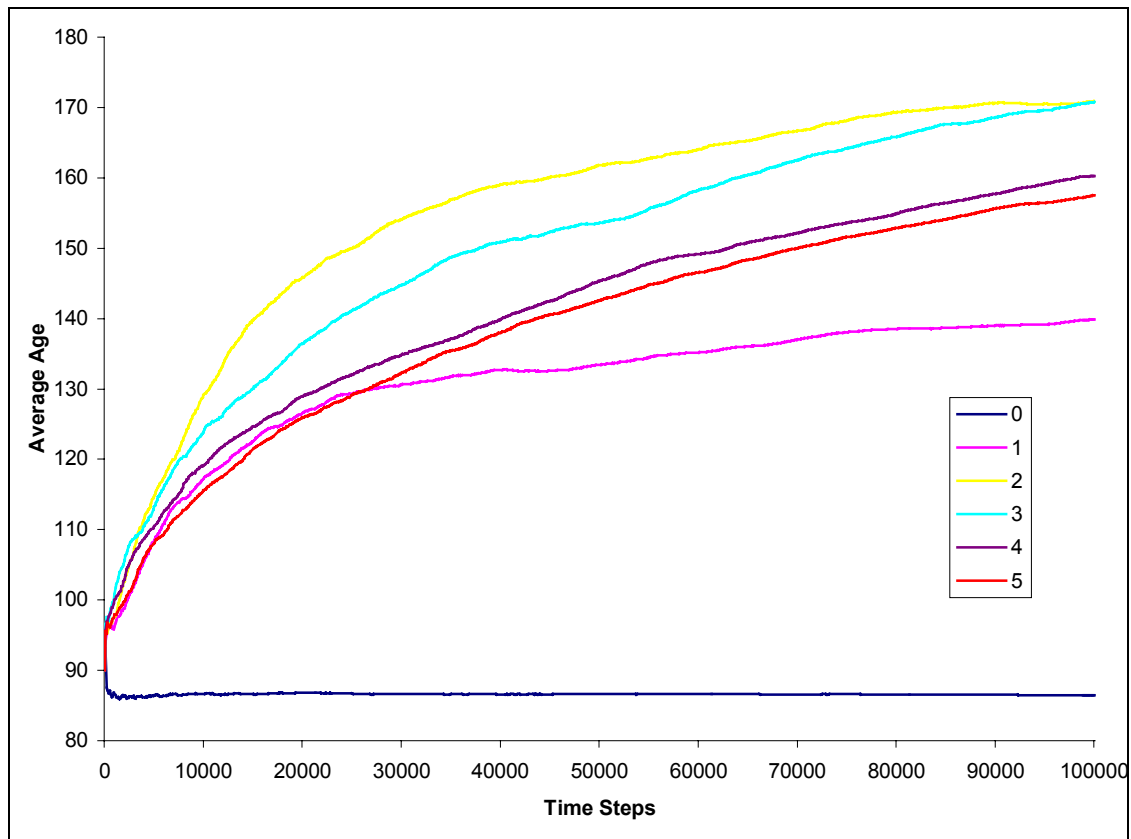


Figure 18: Variations in people's performance with different levels of internal stimuli granularity over 100000 time steps.

When the granularity is set to 0, only one state exists and no comparison can be made between how hungry, thirst, tired, hot and sleep a person is. As can be expected, performance is very poor in this case. Performance is also mediocre with the parameter set to 1. Performance increases as more information is provided to the learning mechanism, by increasing the granularity of the sensed state of the internal stimuli. However, the more information provided, the slower the learning mechanism is to provide good results: at the end of the simulation, the performance with a granularity of 4 or 5 is less than that for a granularity of 2 or 3, but is still increasing steadily.

To investigate this further, the simulation was run for more time steps. Figure 19 is an extension of Figure 18, covering ten times as many time steps for the interesting levels of internal stimuli granularity, plus one extra value for this parameter: 6.

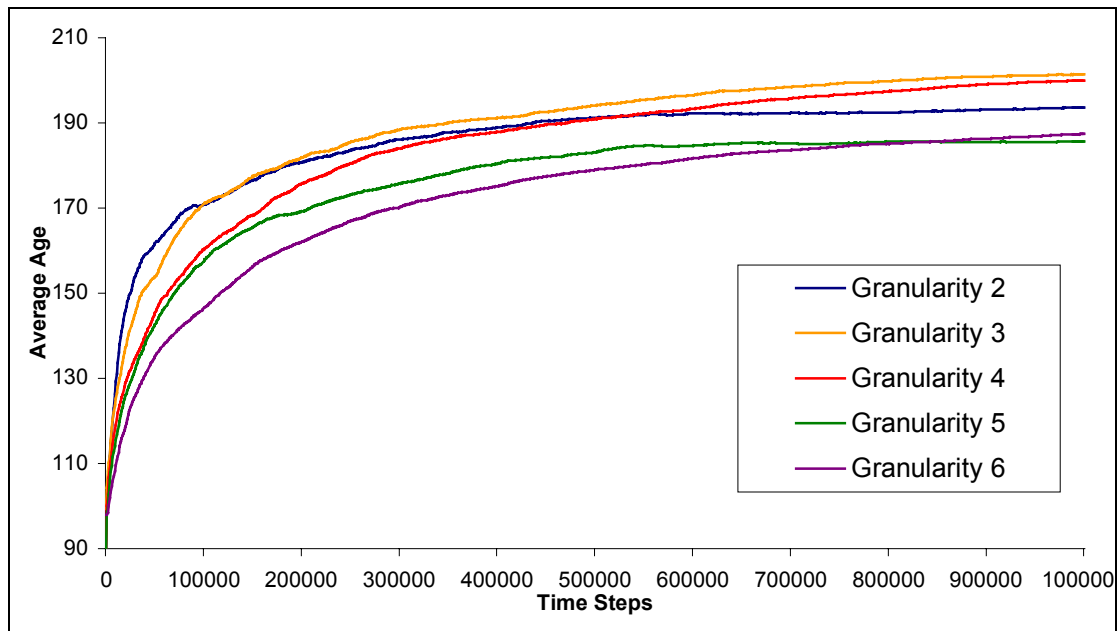


Figure 19: Variations in people's performance with different levels of internal stimuli granularity over 1000000 time steps.

As before, the higher the granularity, the slower the average age increases, in this case, a granularity of 3 gives the highest final average age, but it seems likely that higher values for this parameter may achieve better results if the simulation was run for even longer.

The performance increase gained by giving the people's learning mechanism with more details of the internal stimuli states and more experience to learn from is not really impressive, and still no where near the performance achieved by the pigs. Thus this is unlikely to be the only cause for the difference in performance between the two species.

6.5 Variations of W-learning

The 'maximize the best happiness' variant of W-learning was also implemented, and was used in this section instead of the default 'minimize the worst unhappiness' (see section 5.1.3). The same parameters, optimised in sections 5.3.1 to 5.3.4 were used in an environment with 20 pigs and 20 people. Using the same genetic algorithm parameters as described in section 5.3.5, the best values for the scaling factors for the rewards found in this case are:

- Thirst = 0.08,
- Hunger = 0.96,

- Fear = 0.28,
- Sleep = 0.76,
- Heat = 0.50,
- Fatigue = 0.67.

Figure 20 is a graph of how the people performed using these values.

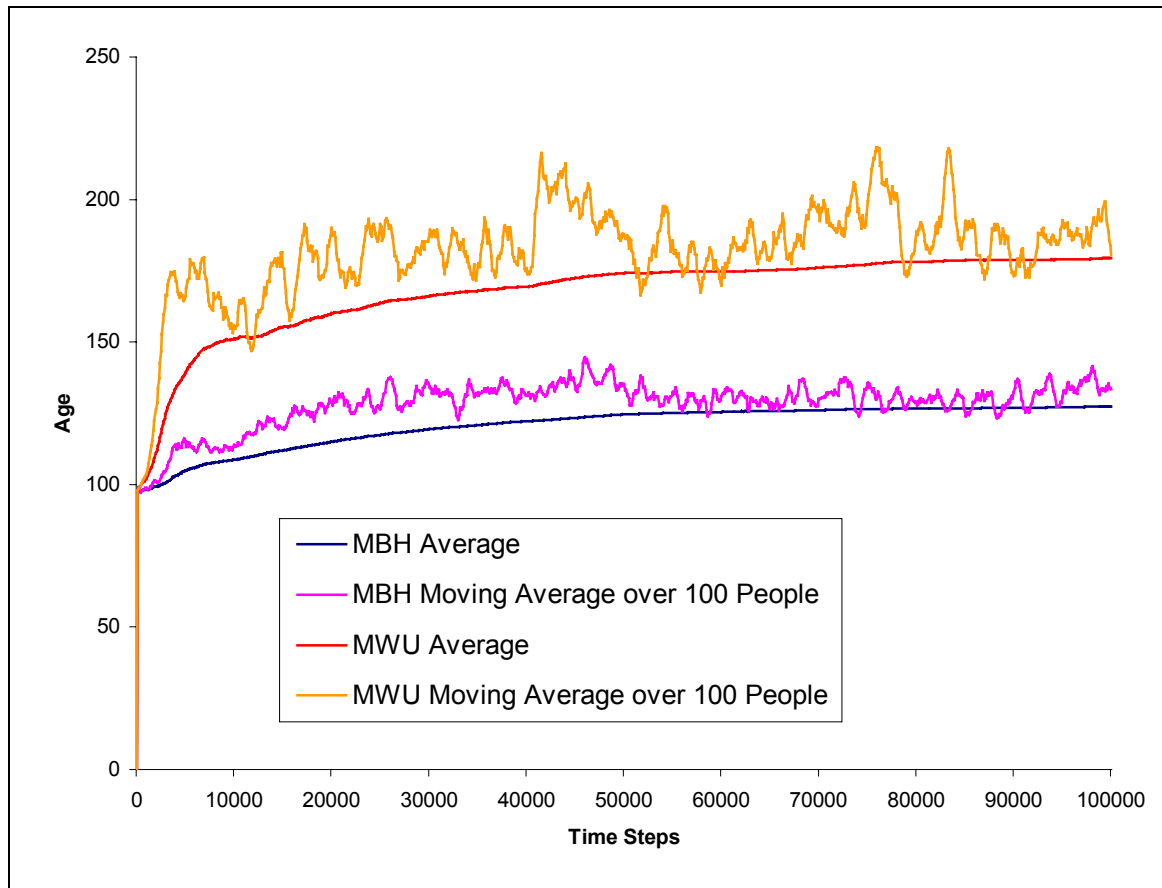


Figure 20: Comparing the ‘minimize the worst unhappiness’ (MWU) and ‘maximize the best happiness’ (MBH) variations of W-learning.

‘Maximize the best happiness’ achieve an average of around only 120 compared to around 170 for ‘minimize the worst unhappiness’. The student t-test, performed over the 6 runs shows that the results from the two variants of W-learning are significantly different, with certainty over 99%. This suggests that the calculation of W values is worthwhile, as ‘maximize the best happiness’ is equivalent to setting $W = Q$. Although ‘maximize the best happiness’ does not perform as well as ‘minimize the worst unhappiness’ it is also successful learning by the definition given in section 5.1.

The ‘minimize the collective unhappiness’ and the ‘maximize the collective happiness’ variants of W-learning were not implemented here.

7 Discussion

What was achieved in this project? What transpired from the work done? What remains yet to be done? We shall now attempt to answer these questions in turn.

7.1 What was Achieved

‘Pigs and people’, a simulated environment with a relatively realistic action selection problem, was created. Two action selection mechanisms were then implemented in the main agents in this environment: the drives action selection mechanism in pigs and W-learning in people. These two mechanisms were then compared under various environmental conditions.

‘Pigs and people’ was written Java, which allows development and execution on most widely used computer platforms. Much care was taken in the development to ensure independence between the action selection mechanisms and the environment. Furthermore, the graphical interface and the terrain engine are completely independent, allowing batch optimisation and evaluation runs, for which the tools are already developed. It would thus be relatively easy for any interested party to implement and test new action selection mechanisms for the pigs or the people.

7.2 What was Discovered

Learning is difficult to implement and rather slow. Furthermore, even once successful learning had taken place, the learning mechanism was still no match for the hand-coded behaviour of the drives mechanism.

W-learning performed significantly less well than the drives action selection mechanism in all experiments. This is not only due to the fact that people need to learn their behaviour from scratch. Indeed, as state-spaces and reward functions had to be designed by hand for the W-learning mechanism, they may have not been optimal. Thus, it could have been impossible for the people to learn optimal performance. Perhaps a different design of state-action spaces and reward functions would have fared better, but it was difficult enough to design those used in this project that did learn something.

In an attempt to improve the people’s performance, more detail was given to their learning mechanism (see section 6.4). With more information available, overall performance improves only slightly, and learning becomes very slow.

7.3 Comparing Drives and W-Learning

Humphrys, [1996b] acknowledges the complexity of designing local reward functions. However, designing the state-action spaces necessary for W-learning was also found to be a challenging task. The state-action spaces may have been more

difficult to design in pigs and people because there are discrete variables to deal with; this is different from the strictly boolean world of Humphrys' house robot. In a boolean environment, there would be no need to introduce and find a reasonable value for the internal stimuli granularity, introduced in section 5.2.1. Designing the drive calculations and the behaviours for the drives action selection mechanism required some thought and tweaking; but substantially less effort was needed for this than for the W-learning implementation.

The drives action selection mechanism does not specify in which way each behaviour, when selected as the one with the highest drive, should choose which action to perform. This choice was a relatively trivial task in this project, as the implementer of the drives mechanism had complete knowledge of the environment. However, should this not be the case, the fact that the low-level actions for each behaviour need to be specified manually may be a stumbling block in an attempt to implement the drives action selection mechanism.

Both the drives mechanism and W-learning were implemented with insider knowledge of the problem. This may have been more to the advantage of the drives mechanism, which could suffer more should the problem be unknown or only partially known. This due to the fact that the choice of low-level actions for a given behaviour in a given state needs to be hand coded by the designer. Of course W-learning also requires the design of state-action spaces and reward functions, and this can be a difficult problem as mentioned earlier. However, Humphrys [1996b] notes that we can add more behaviours than are probably necessary. If they are not useful towards the agent's overall goal, they simply will not be selected, after this is learnt by the W-learning process. This implies that there may be some way to automatically refine the behaviours and their state-action spaces and reward functions, to increase their effectiveness for the agent.

W-learning took far longer to implement, and is by far more complicated than the drives action selection mechanism, containing many more parameters and implementation possibilities. Furthermore, it didn't come close to the success of the drives mechanism, although sensible behaviour was exhibited. However, W-learning may still be the mechanism to choose should the problem to be solved be unfamiliar to the implementer, as W-learning is the more complete action selection mechanism that learns to choose both the behaviour to perform and the low-level actions within that behaviour. Furthermore, there is no guarantee that the parameters, optimised in sections 5.3.1 to 5.3.4, are independent: optimal results may not have been achieved by optimising them individually. A multi-dimensional hill-climbing optimisation with random starts may be the best way to optimise these parameters as the changes in performance seem fairly continuous, but a genetic algorithm could also be used.

7.4 The W-Learning Implementation

Implementing W-learning was an eventful process, a variety of interesting problems arising throughout. The internal stimuli granularity parameter was introduced because of one of these problems, the method used to balance exploration and exploitation had to be changed because of another.

7.4.1 The Internal Stimuli Granularity Parameter

Initially, once a person moved onto a square with some water on it, he would sit on that square and drink and drink and drink... and then die of hunger. After the first few drink actions, the person's thirst level would be reduced to 0, but he would continue attempting to drink as he had learnt that that was the optimal action to perform when on the same square as a water entity. This unforeseen problem occurred in pigs and people, but not Humphrys' house robot simulation [Humphrys, 96a]. The key difference which caused this problem to emerge is that all variables are binary in the house robot problem: either there is a fire or there is not, if the robot just picked up dirt, then there is no more dirt on that square. However, in pigs and people, resources contain a discreet amount of units available, which may not all be consumed in one action.

Thus, the internal stimuli granularity parameter, described in section 5.2.1, was introduced. The reward function for 'drink' was changed so that the reward for drinking when only slightly thirsty is much less than that received when very thirsty. The same was applied for the 'eat' reward function. The reward received is proportional to the amount of resource actually consumed, but the proportionality is mapped into bands, one for each bullet point in section 5.2.2. The higher the internal stimuli granularity, the more bands are used, providing more detail of the actual state-action space values to the learning mechanism. Other possible solutions were considered for solving the 'drink to death' problem:

- Extending the state space and reward functions to incorporate the last action. This seems like cheating, as there seems to be no justification to it. May as well say explicitly "do not consider the drink behaviour if the last action was to drink". The W values should be able to encode this, given the appropriate state-action space.
- Extending the state space to include whether the person is more hungry than thirsty. This goes against the idea that each W-learning table should only be aware of information relevant to it. The full state space could be used, but the reward functions would need to change, and we have seen how large the full state space can become in section 5.2.1.
- Splitting the eating and drinking behaviours into finding resource and consuming resource behaviours. This would still require the internal stimuli granularity parameter, but would reduce the total size of the W-learning tables needed.

Another possible solution would be to use a mechanism to compare stimuli strength directly, thus eliminating the need to encode them in the learning mechanism, which uses discrete states. The drives action selection mechanism compares the values on a continuous scale, which is likely to be one of the reasons it outperforms W-learning. Modifications could be made to the W-learning action selection mechanism to allow it to compare stimuli strength on a continuous scale to help select the appropriate behaviour. However, this may resemble hierarchical Q-learning, with

a manually implemented drives mechanism to select which sub-agent to use instead of the learning switch introduced in section 5.1.2.

7.4.2 Balancing Exploration vs. Exploitation

One way of balancing exploration vs. exploitation in reinforcement learning mentioned in section 5.1.1 is to set the initial Q values higher than they possibly could be, to ensure all actions are tried. It is worth mentioning that this is not an option in W-learning, as W values depend on the Q values. If the initial Q values are too high, then a state can build up an unassailable W value at the start of the learning process. In any case, behaviours may learn W values that do not correspond to the true interactions between the agent and the environment.

This was discovered when the sleep behaviour did exactly that. Optimistic starts were being used to balance the exploration vs. exploitation problem, and the sleep behaviour almost immediately built an unassailable W value. Sleeping is the first thing people learn to do, because on average the sleep level increases faster than any other level when behaving randomly as the people initially do.

Therefore the ϵ -greedy method was used to solve the exploration vs. exploitation in this paper. This is because it is a method that is easy to understand and implement. The softmax method was also implemented using a Boltzmann distribution, but it was not easy to find an appropriate value for the necessary temperature parameter. This was therefore not used in any experiments.

7.5 Further Work

Three possible aspects to focus on when considering further work on pigs and people: improving the action selection problem to make it more realistic, implementing and experimenting with more action selection mechanisms, and improving this implementation. These three aspects are now discussed in turn.

7.5.1 Improving the Action Selection Problem

To improve the action selection problem is to make it more realistic, like the problem faced by living animals in a real world environment. There is obviously much which could be done to achieve this, indeed if it is possible to do so in a computer simulation. Following are some possible ideas for future development.

Mating could be added as a sub-problem to the overall problem of survival, a problem faced by all animals. The number of times successful mating occurred was the fitness measure for the animal in Tyrrell's simulated environment. To simplify things, issues of individuals needing to be of different sex could be ignored at first, allowing any two individual to successfully mate. Care would need to be taken to ensure that species do not die out before having a chance to learn, if a learning action selection mechanism is used. A new individual could receive learnt information not

from a randomly selected person in the environment, but from one of his ‘parents’. This could obviously then be extended by a new mechanism to propagate knowledge from both parents, but the possibilities here are quite extensive and any further discussion on the subject would be far reaching.

Resources could be added at a steady rate, instead of maintaining a constant number of resource entities in the terrain. In this manner, we could attempt to simulate a known biological phenomenon of oscillations in the population numbers of individuals in the species. This of course could only be done if some mechanism for creating new individuals such as that described above were implemented, as there would never be any oscillating if a new creature is created at each time another dies. Furthermore, there would then be a true competition for shared resources. To further enhance the realism, the rate at which resources are added could vary, and even oscillate in such a way as to emulate changing seasons.

There are no **proscriptive sub-problems** in the environment, which is admittedly a shortcoming. The drives action selection problem would fail to perform the best action in the case of a proscriptive behaviour having the highest drive: other behaviours would not be considered, to see which is the best of all non-proscribed actions. However, variations of W-learning, such as ‘minimize the collective unhappiness’ and ‘maximize the collective happiness’, may be able to deal with this more successfully as actions from all behaviours are considered. ‘Minimize the worst unhappiness’ however is likely to suffer from the same problem as drives, as only the behaviour with the highest W value is considered. Originally tar pits were to be introduced into the pigs and people environment to provide a proscriptive problem: the animal could do anything when next to a tar pit except walk into it. However, the tar pits were not implemented in this version due to time pressure.

The problem would also be more realistic if the **sleep** action was effective if only taken in bouts of several hours. Currently animals sleep for one time step at the necessary intervals to keep their sleep level below the fatal limit. However, should the sleep mechanism be changed in a way to make this ineffective, it would be more reasonable for the people to learn to sleep at night, when their vision radius is reduced. Obviously, this would require changes to the state-space and reward functions in the W-learning implementation, and to the manually crafted behaviours in the drives mechanism, and possibly even to the drive calculations. This may be another type of sub-problem to add to those mentioned in section 3.4.3: it would be periodic, but also require a minimum amount of active time to be effective. Furthermore, it would be more realistic to impose ‘enforced sleep’ on animals should their sleep level go beyond a certain threshold, rather than have them die of sleep deprivation as they do in the current implementation.

As pigs and people can move as **fast as lions**, they are only likely to be caught by lions when another stimulus is higher than that caused by the fear of the lion. For example, if a pig is very thirsty, and near a water entity, it may disregard its fear of the lion to go satisfy its thirst. It would however be more realistic to have the lion move faster than its prey can, but also tire out more quickly. This would allow the lion to catch a prey if it got close enough to be able to reach it in a short sprint.

Many other details could be changed to make the problem more realistic: heat levels should increase more during the day than at night, eating may increase thirst, perception should become less reliable for far away objects, all capabilities could be

negatively affected by high deficits in water, food, sleep... Real environments require a lot of work to model.

7.5.2 New Action Selection Mechanisms

Many existing action selection mechanisms would be interesting to implement in pigs and people; following are a few of the more obvious ones.

First, much more needs to be done with **W-learning**. The learning mechanism is complex, and there are many aspects that yet remain to be investigated, as discussed throughout this project. Furthermore, it would be interesting to use W-learning in the pigs, and the drives action selection mechanism in people, to ensure that this gives opposite results to those given in section 6.

Bryson's architecture, **Edmund**, described in section 2.3.3 would be interesting to implement, not only as a state of the art non-learning mechanism, but also due to the fact that the her architecture can incorporate learning. A non-learning version, if its performance is as superior as in Tyrrell's environment, could be used as the reference to beat for any new non-learning mechanisms. It would also be very interesting to implement a version of the architecture that incorporates a learning mechanism.

It would be interesting to compare the drives mechanism with Tyrrell's extended **Rosenblatt and Payton** architecture, to verify whether the results obtained in [Tyrrell, 93a] are reproduced in the pigs and people simulated environment. The extended Rosenblatt and Payton, could also be compared to Edmund, to see if the relative fitness obtained in [Bryson, 00] also holds in this simulation.

All the learning mechanisms described in section 2.4 would also be interesting to implement; not only to give the mechanisms themselves a second evaluation, but to compare and contrast the different mechanisms in a single environment.

7.5.3 Improving this implementation

Given the improvements to the action selection problem mentioned in section 7.5.1, 'pigs and people' could be enhanced to attempt to simulate animal behaviour more realistically. There are very many phenomena observed in real animals that are not simulated in any way in 'pigs and people': grouping into herds, time-sharing, hunting, cannibalism... It would be interesting to see if adequate behaviour could be achieved by hand-coding these behaviours in a non-learning mechanism. Even more interesting would be to see if a learning mechanism could learn these behaviours, and how it would cope in an environment where such behaviour is advantageous.

Of course there needs to be some animal or environmental motivation for these phenomena. A more accurate modelling of dietary needs could cause a need for meat in the agents and thus hunting. If the agents can fend off attacking hunters when in herds given that they spot the predators early enough, then it would may be beneficial for the agents to group into herds and time-share. If there is too much

competition for water, cannibalism could help reduce that competition, assuming catching individuals of the same species is easier than catching those in other species.

The list given is nowhere near exhaustive, and simulating the problems in the environment is very hard to get right. However, with an incremental approach, a progressively more realistic simulation could be achieved, which may help shed light on areas of real animal behaviour have yet to be well understood.

8 Conclusion

This project aimed to compare the performance of two action selection mechanisms: the drives mechanism and W-learning.

A simulated environment, ‘pigs and people’, was created to this effect. The drives action selection mechanism was implemented in the pigs, W-learning was used in the people. The aim of both animals in ‘pigs and people’ is to survive as long as possible, their performance measure being the average age achieved at the end of a simulation run. The two action selection mechanisms were initially optimised on their own, with only one species in the environment. The performance of the two species, and thus of the mechanisms, was measured under a variety of environmental conditions.

Although W-learning successfully learnt from the people’s interactions with the environment, the pigs consistently outperformed the people by a considerable amount. By giving the learning mechanism more detail of the states sensed by the people, W-learning’s performance was slightly increased. However, learning in this case was slower and needed to span more time steps than in any other experiment to achieve better results. Furthermore, the performance was still nowhere near that achieved by the pigs’ drives mechanism. Implementing the learning mechanism was also far more time consuming than implementing the non-learning mechanism. Additionally, W-learning has more parameters that need adjusting, and thus was harder and more laborious to optimise.

Implementing a learning mechanism is much more time consuming than implementing a non-learning mechanism; and if the problem is known, the learning mechanism is unlikely to produce better results. Learning is also rather slow. Even once successful learning had taken place, the learning mechanism was still no match for a hand-coded behaviour of the drives mechanism. Thus, for any commercial purpose of similar complexity to the action selection problem in ‘pigs and people’, one would be likely to be better off choosing the drives action selection mechanism over W-learning, given a choice of the two.

However the field of learning action selection mechanisms is still wide open for plenty of interesting research. There have been many learning mechanisms proposed besides W-learning, and W-learning itself requires a more thorough investigation than that performed in this project. W-learning was implemented in a modular and portable way, this implementation could thus be used to shorten the development time in future artificial life projects, or as a learning tool in any other action selection mechanism.

It would be very interesting to have a round up of learning action selection mechanisms in the same simulated environment, as Tyrrell did for the non-learning mechanisms in [Tyrrell, 93a]. Eventually, learning action selection mechanisms may prove to be more useful, once they are better understood and have more theory and the appropriate tools to back them. Although the learning mechanism’s performance was not impressive, ‘pigs and people’ takes a step in this direction.

9 Appendix A: The Code, and How To Run It

The code for pigs and people was written in Java1.3, and uses Swing components only available in this version. The genetic algorithm toolkit GAJIT was used for the genetic algorithm optimisation of the scaling factors for reward functions introduced in section 5.2.2. A total of around 5000 lines of code were needed for the implementation, split up in the following manner:

- 24% for the graphical interface mostly generated automatically using Forte™ for Java.
- 11% for functional testing and running the simulation in a variety of batch modes, with no interface. These include batch optimisations, running the genetic algorithm, and the experimental runs in this paper.
- 65% for the simulation engine itself, including the environment, the resources, animals and their action selection mechanisms.

Approximately 7% of the lines of code in the simulation engine were for the implementation of the drives action selection mechanism. W-learning required nearly five times as much code: 33% of the total code for the simulations engine.

The simulation can be run in a browser window with the Java Plug-in version 1.3 or later from <http://www.j.uklinux.net/pnp>; or using Sun's appletviewer version 1.3 or later using, for example: 'appletviewer <http://www.j.uklinux.net/pnp>'. A zipped version of the code can also be downloaded from this address, and there is also a link to the complete documentation for the source code generated with 'javadoc'. The code can be compiled using any Java compiler that supports Java version 1.3 and above. The javadoc documentation shows which files to use to run the simulation in batch mode.

GAJIT is available from <http://www.angelfire.com/ca/Amnesiac/gajit.html>.

10 Appendix B: Tables learnt in W-learning

The W table learnt for the drink behaviour will be used as an example in this appendix. A randomly selected drink table after 10000 time steps:

	0	1	2	3	4	5	6	7	8	9	
0	[0.35]	0.11	0.02	0.03	0.20	0.09	0.08	0.17	0.03	0.15	(-13.96)
1	0.00	[0.22]	0.03	0.01	0.00	0.00	0.00	0.00	0.01	0.00	(-20.21)
2	0.00	0.04	[0.17]	0.03	0.00	0.00	0.10	0.00	0.00	0.03	(-21.77)
3	0.03	0.00	0.00	[0.15]	0.02	0.00	0.00	0.00	0.03	0.03	(-13.41)
4	0.00	0.02	0.00	0.02	[0.22]	0.03	0.00	0.00	0.00	0.00	(-12.67)
5	0.00	0.00	0.00	0.00	0.00	[0.13]	0.00	0.00	0.00	0.00	(-10.97)
6	0.00	0.00	0.00	0.01	[0.07]	0.00	0.00	0.00	0.00	0.00	(-1.73)
7	0.02	0.00	0.00	0.00	0.00	0.01	[0.08]	0.00	0.00	0.00	(-26.79)
8	0.04	0.01	0.00	0.00	0.00	0.19	0.01	0.04	[0.20]	0.06	(-14.53)
9	0.00	0.00	0.00	[0.03]	0.00	0.00	0.00	0.00	0.00	0.00	(-53.42)
10	[0.75]	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.27	(12.62)
11	0.00	[0.26]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(-5.84)
12	0.00	0.00	0.00	0.00	[0.09]	0.00	0.05	0.00	0.00	0.00	(-0.04)
13	0.00	0.00	[0.21]	0.00	0.02	0.00	0.00	0.00	0.00	0.00	(-5.31)
14	0.00	0.00	0.07	0.00	[0.44]	0.00	0.00	0.00	0.00	0.00	(2.91)
15	0.00	0.00	0.00	0.00	[0.14]	0.00	0.00	0.00	0.00	0.00	(0.59)
16	0.00	0.00	0.00	0.00	0.00	0.00	[0.42]	0.00	0.00	0.00	(0.66)
17	0.00	0.00	0.00	0.00	[0.02]	0.00	0.00	0.00	0.00	0.00	(-21.89)
18	0.00	[0.05]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(-11.25)
19	0.00	[0.02]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(-55.53)
20	[0.92]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(0.00)
21	[0.00]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(-2.20)
22	[0.00]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(-0.00)
23	[0.00]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(-0.00)
24	[0.00]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(-0.00)
25	[0.00]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(-0.00)
26	[0.00]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(-0.00)
27	[0.00]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(-1.18)
28	[0.00]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(0.00)
29	[0.00]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	(-4.76)

There is one state per row, labelled 0 to 29. There are 30 states because an internal stimuli granularity of 2 was used as described in section 5.2.1. The first ten states are for the 10 possible directions water can be seen from when the internal thirst level is below 33. States 10 to 19 are for an internal thirst level between 33 and 66, and states 20 to 29 are for a thirst level greater than 66. There is one column per action, action 0 is drink, actions 1 to 9 are the possible move actions. Values in squared brackets are the maximal value for that row, and thus the highest Q value for that state. Values in parentheses at the end of each row are the W values for that state, multiplied by a factor of 1000.

The table is organised in such a way that if the maximal Q value in each row is assigned to the direction to go in to actually approach the water, the bracketed values will form three diagonals down the table: from (0,0) to (8,8), from (0, 10) to (8, 18) and from (0, 20) to (8, 28). Rows 9, 19 and 29 are not included in these diagonals as they encode the state where no resource is visible. The code documentation mentioned in appendix A details how the mapping between indices and states and actions.

Following is a randomly selected drink table after 100000 time steps. The table has been learnt better, as the diagonals mentioned above are approximated far better than after only 10000 time steps.

	0	1	2	3	4	5	6	7	8	9	
0	[0.23]	0.07	0.16	0.16	0.04	0.05	0.09	0.05	0.12	0.15	(-9.13)
1	0.03	0.01	0.01	0.02	0.03	0.03	0.00	0.02	[0.13]	0.01	(-4.31)
2	0.03	0.01	[0.16]	0.08	0.08	0.01	0.05	0.02	0.06	0.07	(-3.40)
3	0.01	0.04	0.01	[0.16]	0.01	0.02	0.02	0.02	0.02	0.01	(0.67)
4	0.02	0.00	0.05	0.02	[0.13]	0.06	0.04	0.00	0.05	0.02	(-3.45)
5	0.02	0.00	0.03	0.01	0.03	[0.06]	0.03	0.02	0.00	0.03	(-13.42)
6	0.04	0.07	0.00	0.03	0.06	0.04	[0.13]	0.03	0.02	0.04	(-0.90)
7	0.01	0.00	0.01	0.02	0.00	0.00	0.02	0.01	[0.09]	0.01	(-0.56)
8	0.06	0.06	0.01	0.01	0.01	0.01	0.04	[0.16]	0.05	0.01	(-1.61)
9	0.00	0.00	0.00	0.00	0.00	0.00	[0.00]	0.00	0.00	0.00	(-26.19)
10	[0.69]	0.07	0.00	0.09	0.29	0.21	0.24	0.16	0.26	0.25	(0.51)
11	0.03	0.01	[0.22]	0.00	0.01	0.02	0.00	0.01	0.10	0.02	(-0.77)
12	0.07	0.03	[0.33]	0.10	0.03	0.00	0.22	0.06	0.00	0.01	(0.00)
13	0.01	0.00	0.02	[0.37]	0.02	0.00	0.00	0.00	0.00	0.02	(0.04)
14	0.00	0.00	0.04	0.08	[0.23]	0.00	0.00	0.00	0.00	0.00	(0.58)
15	0.00	0.00	0.02	0.00	0.02	0.00	[0.09]	0.02	0.00	0.03	(-0.00)
16	0.01	0.01	0.01	0.00	0.01	[0.16]	0.01	0.00	0.01	0.02	(0.18)
17	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	[0.06]	0.00	(-0.85)
18	0.00	0.10	0.00	0.01	0.00	0.00	0.11	0.07	[0.34]	0.13	(0.25)
19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	[0.00]	0.00	0.00	(-28.52)
20	[1.38]	0.00	0.40	0.39	0.29	0.00	0.00	0.00	0.00	0.58	(0.35)
21	0.00	0.01	[0.11]	0.01	0.01	0.00	0.00	0.00	0.00	0.00	(-4.69)
22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	[0.02]	0.00	0.00	(0.00)
23	0.00	0.00	0.00	0.00	0.00	0.00	[0.21]	0.04	0.00	0.00	(0.00)
24	0.21	0.00	0.00	0.00	[0.74]	0.00	0.00	0.16	0.00	0.00	(0.00)
25	0.00	0.00	0.00	0.00	[0.43]	0.33	0.00	0.05	0.00	0.04	(-0.00)
26	0.12	0.10	0.06	0.17	0.03	0.14	[0.71]	0.00	0.00	0.12	(0.20)
27	0.08	0.06	0.02	0.00	0.02	0.00	0.02	[0.71]	0.03	0.00	(-1.71)
28	0.03	[0.17]	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.06	(-0.59)
29	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.03	[0.03]	0.02	(-2.30)

Note that the highest W values are in states 10 and 20, which is to be expected as these are the states where a person is thirsty, and currently on the same square as a water entity.

11 Bibliography

[Baerends, 76] Baerends, G. 1976. *The Functional Organisation of Behaviour*. Animal Behaviour, 24, 726-735.

[Blumberg, 94] Blumberg, B. 1994. *Action Selection in Hamsterdam: Lessons from Ethology*. In From Animals to Animats, Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour (SAB-94), Cliff, D., P. Husbands, J.A. Meyer, and S.W. Wilson, eds. MIT Press, Cambridge, Ma.

[Blumberg et al., 96] Blumberg, B., Todd, P. and Maes, P. 1996. *No Bad Dogs: Ethological Lessons for Learning in Hamsterdam*. In From Animals to Animats, Fourth International Conference on Simulation of Adaptive Behaviour (SAB-96). Cambridge: MIT Press

[Brooks, 86] Brooks, R. 1986. *A robust layered control system for a mobile robot*. IEEE Journal of Robotics and Automation, RA-2:14-23.

[Bryson and McGonigle, 97] Bryson, J., McGonigle, B. 1997. *Agent Architecture as Object Oriented Design*. The Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL97)

[Bryson, 00] Bryson, J. 2000. *The Study of Sequential and Hierarchical Organisation of Behaviour via Artificial Mechanisms of Action Selection*. M.Phil Thesis. University of Edinburgh.

[Drescher, 91] Drescher, L. 1991. *Made-up Minds, A Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge, Ma.

[Fikes and Nilsson] Fikes, R and Nilsson, N. 1971. *STRIPS, a new approach to the application of theorem proving to problem solving*. Artificial Intelligence, 5(4):349-371.

[Humphrys, 95] Humphrys, M. 1995. *W-learning: Competition Among Selfish Q-learners*. University of Cambridge Computer Laboratory technical report no.362.

[Humphrys, 96a] Humphrys, M. 1996. *Action Selection methods using Reinforcement Learning*. In From Animals to Animats, Fourth International Conference on Simulation of Adaptive Behaviour (SAB-96). MIT Press, Cambridge, Ma.

[Humphrys, 96b] Humphrys, M. 1996. *Action Selection methods using Reinforcement Learning*. PhD Thesis. University of Cambridge, England.

[Hull, 43] Hull, C. 1943. *Principles of Behaviour: an Introduction to Behaviour Theory*. D. Appleton-Century Company, Inc.

[Lin, 93] Lin, L-J. 1993. *Scaling up Reinforcement Learning for Robot Control*. Proceedings of the Tenth International Conference on Machine Learning.

[Lorenz, 81] Lorenz, K. 1981. *Foundations of Ethology*. Springer-Verlag.

[Maes, 90] Maes, P. 1990. *How To Do The Right Thing*. Connection Science Journal 1(3), 291--323

[Maes, 91] Maes, P. 1991. *A Bottom-Up Mechanism for Behaviour Selection in an Artificial Creature*. In From Animals to Animats, First International Conference on Simulation of Adaptive Behaviour. MIT Press, Cambridge, Ma.

[Mitchell, 97] Mitchell, T. 1997. *Machine Learning*. McGraw-Hill Series in Computer Science. ISBN 0-07-115467-1.

[Möller et al., 00] Möller, R., Lambrinos, D. Roggendorf, T., Pfeifer, R. and Wehner, R. 2000. *Insect Strategies of Visual Homing in Mobile Robots*. In Robotics and Autonomous Systems, special issue: Biomimetic Robots.

[Mott, 81] Mott, D. 1981. *Sensori-motor Learning in a Mobile Robot*. Dept. Comp. Science and Statistics, Queen Mary College, University of London, PhD Thesis.

[Roitblat, 91] Roitblat H. 1991. *Cognitive Action Theory as a Control Architecture*. In From Animals to Animats, First International Conference on Simulation of Adaptive Behaviour. MIT Press, Cambridge, Ma.

[Rosenblatt and Payton, 89] Rosenblatt, K. and Payton, D. *A Fine-grained Alternative to the Subsumption Architecture for Mobile Robot Control*. In Proceedings of the IEEE/INNS International Joint Conference on Neural Networks, IEEE.

[Spier, 97] Spier, E. 1997. *From Reactive Behaviour to Adaptive Behaviour*. University of Oxford, PhD thesis.

[Spier and McFarland, 97] Spier, E. and McFarland, D. 1997. *Learning To Do Without Cognition*. In Proceedings of From Animals to Animats 5: The Fifth International Conference on Simulation of Adaptive Behaviour. MIT Press, Cambridge, Ma.

[Sutton, 90] Sutton, R. S. 1990. *Integrated Architectures for Learning, Planning, and Reacting Based on Approximate Dynamic Programming*. In Proceedings of the Seventh International Conference on Machine Learning. pp. 216-224, Morgan Kaufmann.

[Sutton and Barto, 90] Sutton, R. S. and Barto A. G. 1990. *Time-Derivative Models of Pavlovian Reinforcement*. In Learning and Computational Neuroscience:

Foundations of Adaptive Networks. Gabriel, M and J. Moore, eds. MIT Press, Cambridge, Ma.

[Sutton and Barto, 98] Sutton, R. S. and Barto A. G. 1997. *Reinforcement Learning*. The MIT Press. ISBN 0-262-19398-1

[Tinbergen, 51] Tinbergen, N. 1951. *The Study of Instinct*. Clarendon Press.

[Tyrrell, 92] Tyrrell, T. 1992. *Defining the Action Selection Problem*. In Proceedings of the 14th Conference of the Cognitive Science Society.

[Tyrrell, 93a] Tyrrell, T. 1993. *Computational Mechanisms for Action Selection*, PhD thesis, University of Edinburgh, Centre for Cognitive Science.

[Tyrrell, 93b] Tyrrell, T. 1993. *The Use of Hierarchies for Action Selection*. Adaptive Behaviour, 1, 387-420.

[Tyrrell, 94] Tyrrell, T. 1994. An Evaluation of Maes' Bottom-Up Mechanism for Behaviour Selection. In *Journal of Adaptive Behaviour* 2(4): 307--348.

[Watkins, 89] Watkins, C. 1989. *Learning from Delayed Rewards*. PhD Thesis. University of Cambridge, England.

[Witkowski, 98] Witkowski, M. 1998. *Integrating Unsupervised Learning, Motivation and Action Selection in an A-life Agent*. In Floreano, D., Mondada, F., and Nicoud, J.-D., (Eds.), 5th European Conference on Artificial Life (ECAL-99), pages 355-364, Lausanne. Springer.