Learning Mazes with Aliasing States:

An LCS Algorithm with Associative Perception

Zhanna V. Zatuchna   zhanna.zatuchna@gmail.com

Anthony J. Bagnall   ajb@cmp.uea.ac.uk

School of Computing Sciences, University of East Anglia, Norwich,NR4 7TJ, England

http://www.cmp.uea.ac.uk/Research/kdd/projects.php?project=17

Phone: +44 77 380 80 580

Fax: +44 707 502 3 464

Abstract

Maze problems represent a simplified virtual model of real environments that can be used for developing core algorithms of many real-world application related to the problem of navigation. However, the best achievements of Learning Classifier Systems (LCS) in maze problems are still mostly bounded to non-aliasing environments, while LCS complexity seems to obstruct a proper analysis of the reasons of failure. Also, despite the fact that the maze environment problem has a long history of usage in research into learning, there has been little analysis on the complexity of maze problems. To overcome these restrictions we try to improve our understanding of the nature and structure of maze environments. We analyze mazes used in research for the last two decades, introduce a set of maze complexity characteristics and develop a set of new maze environments.

We then construct a new LCS agent so that it has a simpler and more transparent performance mechanism, and still could solve mazes better than existing algorithms. We use the structure of a predictive LCS model, strip out the evolutionary mechanism, simplify the reinforcement learning procedure and equip the agent with the ability of associative perception, adopted from psychology. We then run our new LCS with associative perception through the old and new aliasing mazes, which represent partially observable Markov decision problems (POMDP) and demonstrate that it performs at least as well, and in some cases better than other published results.

Running Head: Learning Mazes with Aliasing States

Keywords: learning agents; learning classifier systems; associative perception; aliasing; maze.

1   Introduction

The ability to adjust behaviour in complex environments is inherent in all living creatures and is the key reason that humans are effective problem solvers. To bring the behaviour of machine intelligence closer to the human level we need to advance its ability to learn in hard, intricate conditions. Maze problems, usually represented as grid-like two-dimensional areas that may contain different objects of any quantity and various quality, serve as a simplified virtual model of the real world. Mazes may seem to be small and unrealistic compare to the real-world conditions, but their impact on developing of the effective navigating algorithms is invaluable. Their relative simplicity allows us to control the process of learning and trace the behaviour of the learning agent at every stage. At the same time the idea of maze environments includes a virtually unlimited number of graduated complexity levels, enabling researchers to use as simple or as complex environments as they need. These two factors make maze environments a good research paradigm for many navigation-based problems of Artificial Intelligence, from domestic appliance robots and autopilots for the automotive industry to network routing agents and autonomous walking robots for space research.

Learning Classifier Systems (LCS) (Wilson, 1995; Lanzi and Wilson, 1999; Stolzmann, 2000; Bull, 2002) seem to be the research tool of choice in maze environments for the last twenty years. Considering that the number of mazes used in LCS research is significantly larger that the number of mazes used in other research (Bagnall and Zatuchna, 2005a), it is reasonable that LCS own the most promising performance results also. Apart from regular static mazes, LCS have been applied to several advanced variations of the maze problem, including dynamic mazes, noisy environments and multi-objective maze environments (**?**; **?**; **?**). LCS have proved to be a successful class of methods for small sized maze environment tasks (Wilson, 1995; **?**; Bull, 2002; **?**), and the number of new and improved versions of LCS (**?**; **?**; **?**; **?**) is increasing with every single year.

However, despite the fact that LCS have advanced significantly in the last years in terms of development of environmental models (**?**; **?**; **?**), the best LCS's achievements in the maze problems are still mostly bounded to non-aliasing environments. So far mazes with aliasing squares present one of the particularly difficult learning problems for an LCS learning agent. Considering the best available to our knowledge performance, the most impressive results in aliasing environments were achieved by those Learning Classifier Systems which used internal memory (**?**; Lanzi, 1997; Lanzi, 1998; Lanzi and Wilson, 1999) and those that predict next state of the environment (Stolzmann, 2000; Metivier and Lattaud, 2002).

The main goal of the research presented in this paper is to try and bring Learning Classifier Systems closer to successful navigating in complex aliasing maze environments. In order to do that we have to answer the following questions: a) what makes maze problems hard for solving by LCS? b) what internal mechanisms and features may impede LCS from solving certain types of mazes? c) what properties, if adopted by LCS, could benefit them in aliasing maze environments?

Despite the fact that the maze problem has a long history of usage in learning research (Cassandra et al., 1994), especially in the Learning Classifier System literature (Wilson, 1995; Lanzi and Wilson, 1999; Stolzmann, 2000; Bull, 2002), there has been little analysis on the complexity of maze problems. In the research we introduce new metrics for classifying the complexity of mazes based on agent-independent and

agent-dependent characteristics of maze environments. We collected 44 distinct mazes that have been used in 63 different papers in machine learning research in the last decades (Bagnall and Zatuchna, 2005a) and propose a set of maze complexity metrics based on the characteristics of these mazes. In making available a large number of new mazes and software to analyse complexity (Zatuchna and Bagnall, 2005b) we provide a firm basis for the evaluation of the ability of learning systems to solve hard maze problems and for future comparison of different learning algorithms.

While designing our new agent, we analyze the impact of one of the greatest advantages of LCS, their ability to generalize, on the process of learning in maze environments and find that in its present form it is suitable for certain maze types only and even can be detrimental for the process of learning in aliasing maze enironments. We also approach the problem of rewarding actions that will lead to future rewards but in themselves receive none. Based on the conclusions made by this point of research, we develop an alternative LCS agent with a simplified rule production and reinforcement mechanism. We remove all complex structures and evolutionary processes from the agent so we could clearly see how it processes the information while walking through the maze.

To advance our design, we use the psychological theories depicting the processes of perception and association in humans and animals (Lorenz, 1935; **?**) to justify the rules by which our agent processes the information received from the environment. This gives us the name for our system, a Learning Classifier System with Associative Perception, or AgentP for short. We also use the psychological theory of temperament (Pavlov, 1927) to incorporate two different procedural techniques into the learning process. As a result we create two variations of AgentP, Self-Adjusting, which is superficial and flexible, and Gradual, which is careful and rigid, and test them on the existent and new maze environments. We try to find our how the learning modes affect the learning process in maze environments and which approach is better for the purpose.

The rest of this paper is structured as follows: Section 2 gives a very brief background into maze environments, LCS and psychology of learning. Section 3 describes the developed learning model and AgentP, a new Learning Classifier System with associative perception. In Section 4 we provide a survey of RL research with maze environments and describe new ways of measuring performance and classifying maze complexity. In Section 5 we extend the preliminary results for AgentP published in (Zatuchna, 2004b; Zatuchna and Bagnall, 2005a) to demonstrate that AgentP can solve mazes faster and with less memory than other LCS. We also present extensive benchmarking results on a large set of new mazes that will allow for greater ease of comparison for future research. Finally, in Section 6, we discuss future directions of the research and provide conclusions.

2   Background and Related Work

Maze problem as a reinforcement learning task involves learning actions to optimize some objective in an environment. We denote the set of possible actions A, the set of environment states S and the environment at time t as S^t. Maze environments (Wilson, 1995; Lanzi and Wilson, 1999; Bull, 2002) are a commonly used problem domain to estimate LCS performance. A maze is a grid-like two-dimensional area of any size,

usually with a rectangular shape. A maze consists of *cells*. Each cell is an elementary maze item interpreted as a single site. A maze cell may be of many different types. Some may be significant for learning purposes in that they convey a reward (usually called *food* cells), others, such as *wall* and *tree* cells, may mean the cell cannot be occupied by the agent. The agent uses a learning algorithm to form a policy to minimize the steps taken to food based on its ability to perceive the environment and the rewards received. The process begins when the agent is randomly placed in the maze on an empty cell. At each step it performs an action by attempting to move to an adjacent cell. The agent is allowed to move in all eight directions, but only through empty cells. Once an external reward is received (by, for example, reaching a food cell), the agent's position is randomly reset and the task repeated.

The first major problem in learning to solve mazes is how to reward actions that are stage setting, i.e. actions that will lead to future rewards but in themselves receive none. LCS, the most popular class of methods that has been applied to mazes, address this problem through a reward sharing scheme, either a bucket brigade algorithm (Holland and Reitman, 1978) or through a Q-learning type of back propagation (Wilson, 1995). The second major problem in solving maze environments arises when the agent is not able to uniquely disambiguate its environment state (different cells look the same to the agent because of its limited perception). The presence of these *aliasing* cells may lead to a non-optimal behaviour and decrease the agent's performance. Two types of approach to this problem have been adopted in LCS research. The first is to add internal memory to the LCS. This approach has been successfully applied to Wilson's ZCS (Cliff and Ross, 1994; Bull and Hurst, 2001) and XCS (Lanzi, 1997; Lanzi and Wilson, 1999) for maze problems. The second is to alter the basic rule structure to include the following state in addition to the current state and action, then monitor how accurately the two states in the rule match the observed states in the environment. This approach has been used in the Anticipatory Classifier System (ACS) (Stolzmann, 2000; Metivier and Lattaud, 2002; Butz et al., 1999), based of the theory of Anticipatory Behavioral Control proposed by Hoffman (**?**). AgentP combines these two approaches incorporating both the predictive rule structure and a memory mechanism (Section 3.2).

There are several major psychological learning theories that represent different levels of mental activity (Pear, 2001). Based on the purpose of the research we choose two basic principles describing low-level processes of the learning mechanism: *imprinting* and the *laws of organization*. The principles represent the mental processes of primary importance and have never been used in Reinforcement Learning research before.

Imprinting. Imprinting is an especially rapid and relatively irreversible learning process first observed and described by Konrad Lorenz (Lorenz, 1935). In the process of imprinting, distinctive attributes of external objects are copied into an individual's memory and become connected with his behavioural reactions.

Laws of Organization. The focus of the theory is the idea of grouping, which occurs when characteristics of stimuli cause an individual to structure or interpret a visual field or problem as a global construct. The rules of interpretation may take several forms, such as grouping by proximity, similarity, closure, etc. These factors are called the laws of organization and have been explained in the context of perception and problem-solving. In Section 3 we describe how these concepts are implemented in our model.

3 AgentP

In the section we present the Associative Perception Learning model, a new concept for modelling the learning process in autonomous learning agents, and AgentP, a new Learning Classifier System based on this learning model (Zatuchna, 2004b).

3.1 Associative Perception Learning Model

The Associative Perception Learning Model is based on the principles of imprinting and laws of organization. These principles are embodied in AgentP through the two basic concepts:

- during the early period of life an individual absorbs the environment signals as they are perceived (imprinting);

- environment signals, received sequentially, are perceived as a single indecomposable image (laws of organization).

The model approaches the problem of learning through mechanism of associative perception and recognition in a complex environment. The model operates perceptive images where information about the environment state at the initial position of the agent is connected with both information about the environment state at the result position and the agent's action. The model links the state at time t, S^t, the action taken a and the next state S^{t+1} together, creating a single image (Figure 1).

$$\left\{ S^t \xrightarrow{} a \xrightarrow{} S^{t+1} \right\}$$
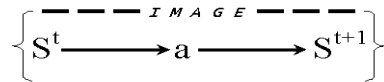
Figure 1: Perceptive image in the Associative Perception Learning Model.

The system employs a refined differentiation mechanism, that allows to provide more precise and accurate recognition of the environment information. Figure 2 shows a functional scheme of the Associative Perception Learning. The set of memorized images (the rule set in AgentP) is altered by the learning algorithm through interaction with the environment, as described in Sections 3.2 to 3.7. The key steps in the associative learning process are as follows:

1. The perceived information is grouped into a single image that includes the two current consecutive environment states S_{cur}^t and S_{cur}^{t+1} associated with the performed action a.

2. The memorized images are in turn compared to the current image. For the performed action a the current initial state S_{cur}^t is compared with the imprinted initial state S_{imp}^t, and the current result state S_{cur}^{t+1} is compared with the imprinted result state S_{imp}^{t+1}.

3. If one state matches and the other does not, the matching state is considered to be aliasing and marked with a sign to allow it to be correctly disambiguated in the future.

4. If there is no exact match for the perceived image, the current image is memorized by imprinting it into memory.

5. If there is an exact match, the parameters of the imprinted image are adjusted by the algorithm described in Section 3.6.
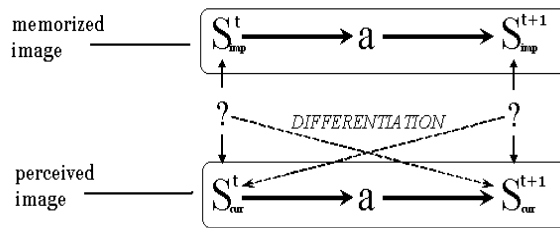
Figure 2: Functional scheme of the Associative Perception Learning.

The associative perception mechanism employs a *sliding image formation* principle (Figure 3). Each representation of an environment state S^t is associated with two others, the previous state S^{t-1} and the following state S^{t+1}. Thus, the formation of images occurs under a sliding state-to-state process. As a result, AgentP is always able to keep track of the connections between images and place the perceived state in the context of the surroundings.
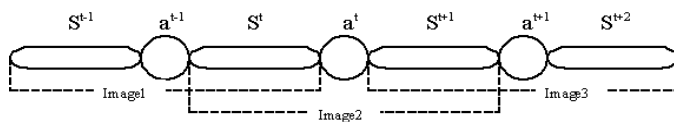

Figure 3: Sliding image formation.

The presented learning model allows us to develop a new psychologically justified algorithm for autonomous learning agents in maze environments. However, the model is quite general and could be adapted for other, non-maze learning tasks.

3.2   Structural Components of AgentP

The basic structure of a single classifier is similar to the *condition* $\rightarrow$ *action* $\rightarrow$ *effect* form used by ACS (Stolzmann, 2000). Unlike ACS, in AgentP consecutive environment states are perceived not only as a cause-effect time vector, but also as a single perceptive image that is compared with previously memorized images for differentiation purposes. By considering both S^t and S^{t+1}, AgentP is able to recognize aliasing in both the current and next states, while ACS is meant to recognize aliasing in the initial state only. In a further deviation from ACS, each classifier is supplemented with an ID system and a verification mechanism. Hence, AgentP could be considered as a hybridisation of basic ACS structure with a memory mechanism, although there are significant differences in performance, reinforcement and learning mechanisms employed.

The parts of the classifier that represent an environment state have an *identification mark, ID* and a *verification sign, F* (see Figure 4). The *ID* is used for refined differentiation of aliasing states. If the verification sign of a state in a classifier is positive, the state is considered as having a *fixed* ID. Each classifier in AgentP is associated with a *prediction d* that represents the expected distance to food at time $t+1$ (Section 3.6). A single classifier in the AgentP model consists of the following components:

initial state S^t with information about attributes of an environment state at time t, where $S^t \in \{0,1\}^n$, and n is the number of the detectors;

$$S \overset{t}{\underline{\hspace{1cm}}} ID_s^t \underline{\hspace{1cm}} F_s^t \longrightarrow a \longrightarrow S \overset{t+1}{\underline{\hspace{1cm}}} ID_s^{t+1} \underline{\hspace{1cm}} F_s^{t+1}$$

Figure 4: Classifier in AgentP.

identification mark ID_s^t for the initial state S^t, where $ID_s^t \in \{0, 1, 2 \ldots \infty\}$;

verification sign F_s^t for the initial state S^t, where $F_s^t \in \{true, false\}$;

action a representing the action undertaken by the effectors at time t;

result state S^{t+1} with information about attributes of an environment state at time $t + 1$, where $S^{t+1} \in \{0, 1\}^n$;

identification mark ID_s^{t+1} for the result state S^{t+1}, where $ID_s^{t+1} \in \{0, 1, 2 \ldots \infty\}$;

verification sign F_s^{t+1} for the result state S^{t+1}, where $F_s^{t+1} \in \{true, false\}$.

AgentP includes a *state list*, L, containing all the states that the agent has perceived during a learning run. Each state in L has a *valency* parameter ν (Section 4.2). Valency reflects the number of aliasing squares included into the state, and, to our best knowledge, has never been used in maze learning research before. In addition, AgentP uses six different classifier sets: the population set $[P]$ which consists of all classifiers; the match set $[M]$ which includes classifiers from $[P]$ that match the environment state S^{t+1}; the initial action set $[A_{ini}]$ which includes classifiers matching the environment state S^t and the action a; the result action set $[A_{res}]$ which includes classifiers matching the environment state S^{t+1} and the action a; the action set $[A]$ which consists of the classifiers matching the current perceptive image I completely; and the previous action set $[A^{t-1}]$ which represents $[A]$ of the previous execution cycle.

3.3 Execution Cycle of AgentP

The execution cycle of AgentP can be summarised as follows.

1. At time $t + 1$ the detectors perceive an environment state S^{t+1} that is explicitly memorized as S_{cur}^{t+1} in the current perceptive image I.

2. State S^{t+1} is placed in the state list L with valency set to 1 ($\nu(S^{t+1}) = 1$), unless it is already present.

3. Classifiers from $[P]$ that match the initial environment state S^t and the performed action a form $[A_{ini}]$.

4. Classifiers from $[P]$ that match the action a and the result environment state S^{t+1} form $[A_{res}]$.

5. The *differentiation* operator (Section 3.4) is applied, and the action set $[A]$ is formed. The specified ID of S_{cur}^{t+1} is put into the perceptive image I where appropriate.

6. The *associative correction* operator (Section 3.5) is applied.

7. Action set $[A]$ becomes $[A^{t-1}]$. Sliding image formation (Figure 3) is performed to update the perceptive image I: $S_{cur}^t = S_{cur}^{t+1}$; a and S_{cur}^{t+1} are cleared.

8. Classifiers from $[P]$ are compared with the new S_{cur}^t to form $[M]$.

9. The *reinforcement learning* operator is applied (Section 3.6).

10. An action for the current performance cycle is selected either by deterministic choice of the classifier with the best prediction D in $[M]$ or, with an exploration probability P_{ex}, by random selection.

11. The action a is performed and memorized in I. A new performance cycle is started.

3.4 Differentiation Operator

Differentiation operates according to the associative perception learning scheme (Fig. 2) and model requirements described in Sections 3.7.1 and 3.7.2. When called, operation is dependent on the valency of the initial state $\nu(S^t)$. If $\nu(S^t)$ is equal to the number of classifiers in $[A_{ini}]$ that have a fixed initial ID, or if S^t was believed to be a non-aliasing state before ($\nu = 1$), the value of the valency ν is increased by 1: $\nu' = \nu + 1$. The current classifier receives a new ID with positive verification sign (fixed ID). If the valency is now $\nu' = 2$, i.e. the aliasing state has just been recognized as an alias, the reference classifier from $[A_{ini}]$ also has its initial ID fixed. The classifier that has increased the valency of a state becomes an *arrow*. Each aliasing cell may have only one arrow and its role is to spread the saved ID through the associative correction operator to all other classifiers that are related to the cell. The same procedure is then applied to the result action set $[A_{res}]$ to differentiate possible aliasing in S^{t+1}. If no new alias has been recognized, a new classifier is created if necessary, but no ID fixing is performed.

For example, being in the state '*0000101000001111*' with current ID identified as 1, the agent performs action '5' and moves to the next square, where its detectors receive the state signal '*1111010111000000*' from the environment. Thus, the perceived image is:

<div align="center">0000101000001111 - 1 - TRUE - 5 - 1111010111000000 - 0 - FALSE</div>

The differentiation operator performs search in the result action set and finds the following classifier:

<div align="center">0000101000001111 - 0 - TRUE - 5 - 1111010111000000 - 0 - FALSE</div>

The classifier looks similar to the perceived image, but has a different ID of the initial state (0 instead of 1). Thus, there are two different initial states, '*0000101000001111-0*' and '*0000101000001111-1*' that lead to the result state '*1111010111000000*' through action '5'. Hence, state '*1111010111000000*' is an alias. To resolve the situation, the ID of the result state of the classifier is fixed:

<div align="center">0000101000001111 - 0 - TRUE - 5 - 1111010111000000 - 0 - TRUE</div>

and a new classifier is created:

<div align="center">0000101000001111 - 1 - TRUE - 5 - 1111010111000000 - 1 - TRUE</div>

The valency of state '*1111010111000000*' is increased by 1. Each of the classifiers corresponds to a separate aliasing square included into the aliasing state '*1111010111000000*'. They both are marked as 'arrow', and each of them will spread it's own ID to the other classifiers related to its square through the associative correction operator.

Figure 5 presents the aforesaid process. The initial state '*0000101000001111*' is marked as A, the result state '*1111010111000000*' is marked as B. The fixed IDs of the classifiers are marked with 'F', the non-fixed IDs are marked with '$-$'.
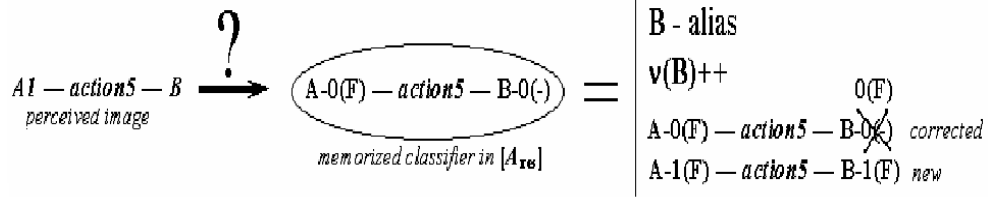

Figure 5: Differentiation of an aliasing state.

3.5 Associative Correction Operator

The associative correction operator performs according to the sliding image formation principle (Figure 3). When the agent is *sure* of its position in the environment (as defined in Section 3.7) the initial state ID of the classifier in $[A]$ can be corrected and verified: $cl(A).ID_s^t = cl(A^{t-1}).ID_s^{t+1}$; $cl(A).F_s^t = true$. The same procedure may be applied conversely. Thus, the associative correction may be performed both ways, from the previous classifier to the following one and backwards. The scheme allows transferring the knowledge about an aliasing state in the classifier population.

For example, the previous action set $[A^{t-1}]$ contains the classifier

$$0000101000001111 - 1 - TRUE - 5 - 1111010111000000 - 1 - TRUE$$

and the current action set $[A]$ contains the classifier

$$1111010111000000 - 0 - FALSE - 4 - 1010101010111010 - 0 - FALSE$$

then the classifier from the previous action set $[A^{t-1}]$ can transfer the fixed ID of its result state to the initial state of the classifier in the current action set $[A]$. After the correction, the action set classifier looks as follows:

$$1111010111000000 - 1 - TRUE - 4 - 1010101010111010 - 0 - FALSE$$

3.6 Reinforcement Learning Process

Analysis of maze complexity characteristics suggests that distance to food plays a significant role in the process of maze learning (Section 4.1.2) because of its connection to the reinforcement process. The Q-reinforcement procedure (Watkins, 1992) has been widely used in LCS (Wilson, 1994; Wilson, 1995; Metivier and Lattaud, 2002). However, the mechanism by which the Q-coefficients depend on the distance to food d may lead to some disadvantages. Considering that $\lim_{t \to \infty} Q = \gamma^d$ (where γ is the discount factor), the expected difference between two successive Q-learning coefficients approaches zero: $\lim_{d \to \infty} (\gamma^d - \gamma^{d+1}) = 0$. The influence of some stochastic processes in the agent may lead to the situation when the actual coefficients Q^A may significantly fluctuate around their rated values Q^R, and upon increasing d the actual coefficients for non-optimal actions may become larger than the actual coefficients for the optimal actions: $|Q^R - Q^A| \geq \gamma^{d-1} - \gamma^d$.

As it is shown in Section 4.2.1, distance to food is one of the crucial factors in aliasing mazes. Because of this, the reinforcement procedure for AgentP has been modified so that the difference between the learning coefficients remains the same with increased distance to food. Thus, instead of Q-learning coefficients, the distance-modified learning coefficients d_{exp} reflecting the expected distance to food from the square are used, such that for any (S^t, a, S^{t+1}) with an expectance to be within d steps to food, the distance coefficient is $d_{exp}(S^t, a, S^t) = d$.

For each performance cycle, when the previous action set $[A^{t-1}]$ contains a single classifier and the initial state of the classifier represents either a non-aliasing state or has a fixed ID, the reinforcement operator is applied. The new value of the prediction d_{exp} for the classifier in $[A^{t-1}]$ is equal to the minimal distance prediction d_{exp} from the current match set $[M]$ increased by 1: $cl(A^{t-1}).d_{exp} = min(d_{exp}(S', a')+1, \forall a \in [M])$. If the minimal distance prediction in the current match set $[M]$ is equal or larger than the prediction of the classifier in the previous action set $[A^{t-1}]$, both the prediction coefficients of the classifiers in the match set and the prediction of the classifier in the previous action set are discarded and reset to the initial value.

To sum it up, the learning coefficient d_{exp} of a classifier reflects the minimal number of steps from the current square to the food cell if the action a advocated by the classifier is used. Thus, unlike in Q-learning, in the distance based reinforcement process the expected difference between two successive learning coefficients remains the same with increased distance to food.

In the present form the reinforcement learning operator introduces certain limitations as the agent will only be able to handle specific kinds of reward function. First, the environment should provide discrete background, which is acceptable for grid-like maze problems, but cannot be directly transferred to a continuous environment. Second, the agent is not designed to operate on multi-motivational tasks, i.e. in maze environments with more than one kind of food or in presence of 'enemies' and/or other agents. The limitation can be overcome at the later stages of the research. The reinforcement procedure in its present form, however, provides simple and reliable mechanism of spreading knowledge in the classifier population and sufficiently serves its main purpose, providing test facilities for the operators of refined differentiation of aliasing squares.

3.7 Learning Models

The learning operators of AgentP are preformed only when the agent is certain about its position in the environment. We denote the state at time t as $D(t)$ if it is distinct and $A(t)$ if it is aliased. At any given time $t+1$ the agent can find itself in one of the following situations regarding certainty of the last three last states, $t-1, t$ and $t+1$:

$D(t-1) \rightarrow D(t) \rightarrow D(t+1)$ - all three states are distinct;

$A(t-1) \rightarrow D(t) \rightarrow D(t+1)$ - state at time $t-1$ is aliasing, the others are distinct;

$D(t-1) \rightarrow A(t) \rightarrow D(t+1)$ - state at time t is aliasing, the others are distinct;

$D(t-1) \rightarrow D(t) \rightarrow A(t+1)$ - state at time $t+1$ is aliasing, the others are distinct;

$A(t-1) \rightarrow A(t) \rightarrow D(t+1)$ - state at time $t+1$ is distinct, the others are aliasing,

$D(t-1) \rightarrow A(t) \rightarrow A(t+1)$ - state at time $t-1$ is distinct, the others are aliasing,

$A(t-1) \rightarrow A(t) \rightarrow A(t+1)$ - all three states are aliasing.

In all cases except the last the agent has at least one certain point (a distinct state) to proceed from. The most crucial moment in the learning strategy is the behaviour of the agent in $A(t-1) \rightarrow A(t) \rightarrow A(t+1)$

situation, i.e. when it is passing through only aliasing squares.

The problem of learning in an uncertain situation can be approached in several different ways. While experimenting with the agent architectures, we try and develop a justified concept of the changes to be introduced. The concept is based on the idea of mobility developed by Pavlov (Pavlov, 1927), a characteristic of the nervous system that is directly connected with the quality of learning process and reflects the adaptive capabilities of an individual. Based on the psychological phenomenon, we introduce two alternative procedural techniques to the learning process that result in creating of two variations of AgentP of 'different temper'. The first, *Self-Adjusting* AgentP, is more flexible and adapts rapidly to changing information; the second, *Gradual* AgentP, is more conservative in drawing conclusions and rigid when it comes to revising strategy.

For any uncertain situation, if the previous or current action set contains more than one classifier, the situation is considered as 'unsure' and no operations are performed. This rule is valid for both the Self-Adjusting and Gradual modes. However, when the previous and the current action set both contain only one classifier each, the decision on whether to call the update operators is different for the two types.

3.7.1 Self-Adjusting Learning Mode

When the previous and current action set contain one classifier each, Self-Adjusting AgentP is thought to be 'sure' about its location and attempts to perform the learning operators with no constraints. The agent increases valency of any state as necessary according to the differentiation operator described in Section 3.4, and all IDs can be transmitted simultaneously. Incorrect IDs are corrected by the rule of trust: an arrow always transmits its ID and a distinct-based classifier always transmits ID to any other conflicting classifier (unless the classifier it is attempting to transmit to is an arrow).

Under these conditions the agent performs as a rapidly self-adjusting system: IDs are immediately transmitted with no precautions and mistakes are adjusted for without checks. This means AgentP can explore all aliasing squares at the same time, but also means that incorrect information may be transmitted. For example, let us assume that the previous action set contains classifier A:

0000101000001111 - 0 - FALSE - 5 - 1111010111000000 - 0 - FALSE,

the action set contains classifier B:

1111010111000000 - 2 - TRUE - 6 - 0000000111000000 - 0 - FALSE,

and all three states that are present in the classifiers, '*0000101000001111*', '*1111010111000000*' and '*0000000111000000*' are aliasing. Classifier A does not have a fixed ID in its initial part, hence, the agent is not sure which particular square of the aliasing state '*0000101000001111*' it came from. Classifier B does not have a fixed ID in its result part, hence, the agent is also not sure which particular square of the aliasing state '*0000000111000000*' it just entered. Despite these facts, the Self-Adjusting learning mode allows the transmission of ID of the aliasing state '*1111010111000000*' from classifier B to classifier A. Thus, after the associative correction operator (Section 3.5) is applied, classifier A looks as follows:

0000101000001111 - 0 - FALSE - 5 - 1111010111000000 - 2 - TRUE.

Let us assume that later in the learning run classifier A is again found in the previous action set, while the current action set contains classifier C:

1111010111000000 - 1 - TRUE - 3 - 1111111111011111 - 0 - FALSE.

The classifiers have conflicting IDs of the state '*1111010111000000*' ($ID = 2$ vs. $ID = 1$). However, if the result state '*1111111111011111*' of classifier C is non-aliasing, the classifier is considered as more trustworthy and transmits its ID to classifier A. After the correction, classifier A looks as follows:

0000101000001111 - 0 - FALSE - 5 - 1111010111000000 - 1 - TRUE.

3.7.2   Gradual Learning Model

Gradual AgentP is only 'sure' about its position in the environment when, firstly, it has been 'sure' on every time step since the last non-aliasing square and, secondly, when at least one classifier in the previous and current action set has a fixed ID for both S^t and S^{t+1}. When performing the operator, the agent cannot increase valency of an aliasing state until the square with the previous ID of the state has been correctly identified. In addition, there is no direct correction of mistakes; if a state that was previously considered as a reliable non-aliasing square has been freshly discovered to be an alias, all classifiers that have IDs of neighbouring cells fixed based on the unreliable information, are deleted. Valency of aliasing states is temporarily decreased if necessary.

The agent uses its fixed classifiers as a path to orient itself in aliasing surrounding. Under the Gradual settings AgentP is a cautious learning system that explores the aliasing environment gradually, building up a consecutive bridge from reliable non-aliasing squares through an aliasing conglomerate. In the example, considered in the previous section, neither classifier A nor classifier B satisfy the requirements of the Gradual learning mode, hence the associative correction operator would not be applied. Let us assume that the previous action set contains classifier D:

1111010111000000 - 1 - TRUE - 1 - 0101010101010101 - 3 - TRUE,

the action set contains classifier E:

0101010101010101 - 0 - FALSE - 2 - 0000101000001111 - 0 - FALSE,

and the agent is 'sure' about its location (in other words, the agent used only classifiers with fixed IDs for all aliasing squares it walked through since the last non-aliasing square). In this case, the Gradual AgentP would apply the associative correction operator, and classifier E would be modified as follows:

0101010101010101 - 3 - TRUE - 2 - 0000101000001111 - 0 - FALSE.

3.8   Generalization and Scalability

Most Learning Classifier Systems (Holland and Reitman, 1978; Wilson, 1994; Wilson, 1995; Stolzmann, 2000) include a generalization mechanism (Section 4.3). The ability to generalize makes an algorithm more scalable and useful, but only if the agent is able to learn generalizations in large problems without a proportional increase in the number of rules required. LCS with generalization that have been used on mazes have tended to have rule sets two orders of magnitude larger than the number of states in the maze. For example, XCSMH needs 6000 rules to solve Woods102, which has 28 different states (Lanzi and Wilson, 1999) and ACS requires 2800 rules to solve maze E1, a 45 state problem. To our knowledge, an experimental study of how rule set sizes for ZCS-, XCS- and ACS-based Learning Classifier Systems increase with the number of states and the

type of aliasing has not been conducted, but the indications are that they may require prohibitively large rule sets to solve larger mazes.

The main benefit of generalization (aside from the discovery of true underlying relationships) is the ability to store information about the environment compactly, thus decreasing the time spent updating the classifier. However, this potential benefit becomes a hindrance if massive rule sets are required to find good generalizations. The larger sets of rules demand not only large amount of memory, but also significantly larger amount of time to converge. For example, Bull (Bull and Hurst, 2002) reported ZCS was able to reach the optimal performance on simple Woods1 maze environment after 10,000 runs. A neural variation of XCS by O'Hara (O'Hara and Bull, 2005a) needed 3000 trials to solve non-aliasing Woods2. The Neural Learning Classifier System with Self-Adaptive Constructivism (Bull and Hurst, 2003) showed the optimal performance on small aliasing Woods101 environment (Fig. 10(a)) after 25,000 problems in a row. Thus, the present approach to generalization, when a learning agent tries to evolve the optimally generalized rules from the beginning, before it actually has learnt the environment, may lead to huge rise in the demand for computational resources.

Unlike other LCS, AgentP does not include any generalization technique. It operates explicitly imprinted states of the environment. The feature is justified by the Associative Perception Learning Model employing the principle of imprinting, emphasizing the fact that during the early period of life an individual absorbs the environment signals as they are perceived. One of the positive things about the algorithm is its minimal memory requirements: for any aliasing maze that is solvable by AgentP it will create the same number of classifiers as it would do if the maze was non-aliasing. More precisely, AgentP will assign a unique ID for any aliasing square, converting the maze into a non-aliasing kind in its own memory. Thus, as the number of cells in a maze goes up, the number of rules required by AgentP to solve the maze adjusts to a value that is directly proportional to the number of cells, duplicated by a coefficient, depends on the maze structure. Thus, the memory required for solving a maze is usually a value of the same order as the size of this maze. For example, the test results discussed in Section 5 show that AgentP requires only 82 rules to solve Woods102, compare to 6000 required by XCSMH (Lanzi and Wilson, 1999), and only 240 rules to solve E1, compare to 2800 rules by ACS (Metivier and Lattaud, 2002).

As for the time required for learning, it will grow in geometric series while maze increase. This could become detrimental in more extensive mazes that include dozens of thousands squares. However, as shown in Section 5, AgentP still outperforms the other agents in terms of learning time. It needs around 25 trials to solve aliasing Woods101 environment, compare to 25,000 required by the classifier system presented in (Bull and Hurst, 2003), and only 22 trials on average to solve MazeF4, while ACS (Stolzmann, 2000) has been reported to need more than 500 trials.

The algorithm presented in this paper is designed as a first stage of maze learning research. The idea of this stage of the research is to sharpen the learning ability of learning classifier systems in maze environments and create a system capable of differentiating the whole range of complex aliasing patterns, from simple to the most sophisticated. The problem of decreasing learning time and creating more compact representation of larger maze environments will be addressed at the next stage of the research. The rough idea for further investigations is to design a new kind of generalization system that could extract a valuable knowledge

from past experience and apply it to the future learning steps. This idea of post-learning generalization mechanism is based on the behavioural phenomenon termed *stimulus generalization* that was first described and interpreted in (Pavlov, 1927). According to the research, an individual that has learnt a certain behaviour, responds in a similar manner to stimuli that are similar to the one on which he was trained. In terms of maze learning, development of a post-learning generalization mechanism would allow the agent to extrapolate the knowledge obtained in one area of the environment to the others areas of the same environment, or, alternatively, the knowledge obtained in a certain environment to be transferred to other similar environments. Thus, being placed in a vast maze environment, the system would start from exploring the immediate surrounding, extract valuable information from its present classifiers into a more compact representation and then use the extraction at the next stages, repeating the procedure continuously. A similar idea was recently expressed by Will Browne and Dan Scott (Browne and Scott, 2005), who proposed an algorithm of *abstraction*. Abstraction attempts to find patterns in the rules that performed best within an LCS agent. Having found a pattern common to two or more rules, the abstraction algorithm generates a new rule based solely on this pattern. The abstraction represents one of possible post-learning generalization mechanisms and seems to be a highly perspective direction for future developments of autonomous learning agents.

4 Maze Problems

Maze environments are multi-step decision processes with rewards only on transitions to terminal states. A maze can be classified by whether it possesses the Markov property (Sutton and Barto, 1998). Non-aliasing mazes can be fully observed by an agent with limited perceptive power. A state signal in a non-aliasing maze succeeds in retaining all relevant information for successful resolving the maze by the agent. Any action taken in any state of the environment always leads to the same next environment state. Thus, non-aliasing mazes satisfy the Markov property and are MDP environments. Aliasing mazes (Section 4.2) are the kind of reinforcement learning tasks where the agent with limited perceptive power cannot distinguish all possible squares. From the agent's point of view it looks like the same action performed in the same state may lead to a different next environment state. Presence of the aliasing squares in a maze is reflected in the characteristics of its transition matrix that describes the probability of moving from one state to another for any given action. Mazes with no aliasing squares have the characteristic that for any state action pair, there will be one state with a probability of transition of 1. If there are at least two aliasing squares then the transition matrix will contain probabilities other than 0 or 1. However, the transition matrix of aliasing mazes remains the same at any step of the time. The agent is said to suffer from the *hidden state* problem. Despite the fact aliasing mazes are frequently called 'non-Markov' in the literature (Bull, 2002; Butz et al., 1999; Cliff and Ross, 1994; Lanzi and Wilson, 1999), they are Partially Observable MDP (POMDP) environments in fact, because they are domains in which actions have probabilistic results (Cassandra et al., 1994; Littman, 1995). Dynamic mazes (Section 4.1) as well as uncertain environments, which include noise (Section 4.3), represent a different kind of learning problems. They do not satisfy Markov property and have a transition matrix that depends on time. Unlike aliasing mazes, dynamic and noisy mazes cannot be turned into MDP by improving the agent's structure and abilities. Dynamic and uncertain environments represent non-Markov

Table 1: Statistics on usage of mazes in research

Mazes	24	11	3	6
Number of research papers	1	2-3	4-5	6 >

Table 2: Statistics on amount of mazes used by a single research

Research papers	33	13	12	5
Number of mazes	1	2	3-4	5 >

reinforcement learning tasks and are excluded from this stage of research.

The techniques applied to mazes can be categorised as follows: XCS: 14 mazes in 20 papers, e.g. (Wilson, 1995; Lanzi, 1998; Lanzi, 1997); ZCS: 8 mazes in 11 papers, e.g. (Wilson, 1994; Cliff and Ross, 1994; Bull, 2002); ACS: 18 mazes in 13 papers, e.g. (Stolzmann, 2000; Butz et al., 1999; Metivier and Lattaud, 2002); Other methods (such as Neural Learning Classifier Systems (Bull and Hurst, 2003; O'Hara and Bull, 2005a), First Visit Profit Sharing algorithm (Arai and Sycara, 2001), etc.): 25 mazes in 19 papers. A more detailed description of research that have been used in maze environments in the last two decades can be found in (Zatuchna, 2004a); the full collection of mazes is available at (Zatuchna and Bagnall, 2005b). Unfortunately, it is unusual for different techniques to use the same mazes. This makes it hard to compare the learning abilities of different algorithms.

Table 1 presents statistics on how many times each maze has been used in different research. It can be seen that the majority of mazes have been used in a single paper only (24 mazes, or 48%), and 35 of the 44 mazes (70%) have been used in at most three publications. This shows that different researchers mostly have used different mazes and this makes it hard to compare the performance of different learning agents based on their maze test results. Table 2 presents information on how many mazes each publication used. We can see that the results for the majority of the research were based on a single maze only (33 papers). 58 of the 63 papers were tested on at most four mazes. Hence the majority of research in this field is evaluated on a small number of mazes, and usually the mazes used have not been used by other researchers.

Thus, we have collated mazes used in published research to allow for more commonality of problem domain across different research areas. A further barrier to comparing learning methods is the lack of a clear understanding of what makes a maze problem complex. The characteristics of mazes that determine the complexity of the learning task fall into two classes: those that are independent of the learning algorithm, such as size and density of obstacles; and those that are an artifact of the agents ability to correctly detect its current state, such as the number and type of aliasing cells.

4.1 Agent independent maze attributes

4.1.1 Size

The number of cells in a maze, s_m, obviously affects the complexity. The mazes range from 18 cells (Cassandra4 (Cassandra et al., 1994), to 1044 cells (Woods7 (Wilson, 1994)). Mazes smaller than 50 cells are

classified as *small* (19 mazes). *Medium* mazes, such as Maze10 (Fig. 14), have between 50 and 100 cells, and *large* mazes have more than 100 cells. Here and further we choose the boundaries of the parameters based on the distribution of the parameters in the collected mazes.

4.1.2 Distance to food

The average distance to food (ϕ_m) is an important characteristic of complexity, because the further the average distance to food is, the harder it is for the reward sharing mechanism to reward stage setting actions. The range of values in the mazes varies from $\phi_m = 1.29$ for Koza92 maze to $\phi_m = 14.58$ for Nevison-maze3 (Nevison, 1999). We classify a maze as having an *extra-short* distance to food if $\phi_m \leq 3$, a *short* distance if $3 < \phi_m \leq 5$, a *medium* distance if $5 < \phi_m < 10$ and a *long* distance if $\phi_m \geq 10$.

4.1.3 Obstacles

Mazes may contain walls, partitions or both. A wall is a complete cell that the agent cannot occupy or see through, whereas a partition is a barrier between cells. For example, Russell&Norvig maze shown in Figure 6(a) (Russell and Norvig, 1994) is a *wall maze*, Gerard-Sigaud-2000 shown in Figure 6(b) (Gerard and Sigaud, 2001) is a *partition maze*, and MiyazakiC (Miyazaki and Kobayashi, 1999) is a *wall-and-partition* maze. Mazes like E2 (Figure 7(b)), that contain only surrounding walls, are *empty* mazes. The number of obstacles in a maze, o_m, is defined as the total number of internal wall cells plus the total number of partitions plus half the total number of surrounding walls. Thus, for mazes with a surrounding wall, we adjust the number of obstacles to allow for the fact that they may only ever present an obstacle from a single direction.

4.1.4 Density

The density of a maze is the proportion of obstacles to squares, $\delta_m = \frac{o_m}{s_m}$. A maze is *spacious* when $\delta \leq 0.5$ and a maze is *restricted* when $\delta \geq 0.7$. Mazes with intermediate values of δ_m are mazes of *average density*. Table 3 gives the distribution of mazes by aforesaid metrics.

Table 3: Distribution of mazes and papers by physical characteristics. The figure in brackets is the number of publications using mazes in each category

size	distance to food			density		
	extra-short/short	medium	long	spacious	average	restricted
small	19 (34)	0	0	0	9 (24)	10 (13)
medium	18 (25)	4 (18)	0	2 (2)	16 (28)	4 (15)
large	2 (16)	1 (0)	6 (2)	3 (7)	6 (10)	0

4.1.5 Geometric form.

Toroidal mazes are mazes without borders, i.e. mazes with closed-loop sides. *Flat* mazes are bounded mazes with surrounding borders. The type of geometric form influences density, as toroidal mazes usually have a lower δ value than similar flat mazes.

4.1.6 Type of objects.

In addition to a target (food) state, some mazes contain a penalizing state, such as *enemy*. For example Russell&Norvig maze (Figure 6(a)) has an enemy marked as E. Enemy and enemy+food mazes present a different learning problem to food mazes, and have been used only in (Russell and Norvig, 1994) and (Littman, 1995). Mazes may also have different types of obstacles as well as different kinds of food, for instance Woods2 (Wilson, 1995). The number of types of object affects the agent's ability to perceive its environment, and hence influences the number of aliasing squares.

Figure 6: (a) Russell&Norvig, $o_m = 10, \delta_m = 0.48$; (b) Gerard-Sigaud-2000, $o_m = 12, \delta_m = 0.71$

4.1.7 Maze dynamics

Some mazes involve cells which change state. For example, a multi-agent maze will have cells that are sometimes empty and sometimes occupied. Other mazes such as Sutton98 (Sutton and Barto, 1998) may include moving walls. On the whole, we can talk about three sources of non-static mazes: dynamics of indifferent objects (walls), dynamics of principal objects (food/enemy) and multi-agent systems. Dynamic mazes are not included into the research.

The complexity of the learning problem is only partially dependent on the physical complexity of the maze. Perhaps of greater importance is the ability of the agent to perceive the environment.

4.2 Agent dependent maze attributes

An agent's ability to perceive its current location in the maze can effect the complexity of the learning problem. The agent may not be able to distinguish one state from another despite the fact that they are in different locations. Cells that appear identical under a particular detector are commonly called *aliasing cells* (or *aliasing squares*), while an environment signal that correspond to two or more aliasing cells is called *aliasing state*. Thus, two or more aliasing squares may appear as a single aliasing state to the agent. We introduce a new parameter ν called *valency*. The valency of a state reflects the number of squares that correspond to that state, so a non-aliasing state will always have $\nu = 1$, and an aliasing state will have $\nu > 1$. A maze containing at least two aliasing cells, is called an *aliasing maze*. Aliasing mazes deserve special emphasis in the context of maze classification because they represent the most difficult class of problem.

(Wilson, 1991) proposes a scheme to classify reinforcement learning environments with respect to the sensory capabilities of the agent. If an environment belongs to Class 1 the agent is able to determine the entire state of the environment. In Class 2 environments the agent has only partial information about the true state of the environment. Wilson consider Class 2 environments to be partially observable with respect to the agent and non-Markov with respect to agent's actions.

(Littman, 1992) presents a more formal classification of reinforcement learning environments, based on the simplest agent that can achieve optimal performance. Two parameters h and β characterize the complexity of an agent. An (h, β) environment is best solved by an (h, β) agent that uses the input information provided by the environment and at most h bits of local storage to choose the action which maximize the next β reinforcements. Hence, Class 1 environments correspond to $(h = 0; \beta = 1)$ and $(h = 0; \beta > 1)$ environments, while Class 2 environments correspond to $(h > 0; \beta > 1)$ (non-Markov) environments. Whilst this classification is useful, there is still a large degree of variation in complexity within Class 2 problem and the nature of the aliasing may alter the difficulty of the learning problem.

4.2.1 Alternative types of aliasing.

The complexity of the maze cannot be determined from just the transition matrix. Some mazes, such as Woods2 (Wilson, 1995), may produce a transition matrix with uncertainty but still be easily solved by a memoryless agent and are, according to (Littman, 1992), Class 1 environments. Thus, the complexity of the problem is determined by not only the uncertainty, but also the optimal strategy. Woods2 is classified as Class 1 because the optimal strategy in the squares that appear the same is identical.

The complexity of a maze for an LCS agent with a particular detector can be quantified by how long, on average, an agent using a Q-table trained by the Q-learning algorithm takes to find food compared to the optimal steps to food. If Q-learning can disambiguate all squares then, assuming it has been trained for long enough, it will find the optimal route to food. If, however, it has a detector that introduces aliasing, it will take longer if the aliasing affects the optimal strategy. We use a standard version of the Q-learning algorithm, $\gamma = 0.2, \alpha = 0.71$, with roulette-wheel action selection in exploration mode and greedy action selection (max Q) in exploitation mode (which is every other time), number of trials $n = 20000$. Let ϕ_m^Q be the average steps to food of a trained Q-learning agent that can only detect the surrounding squares. The complexity measure ψ_m is then defined as $\psi_m = \frac{\phi_m^Q}{\phi_m}$. This measure gives us a metric that can quantify the effects of aliasing. For example, mazes E2 (Metivier and Lattaud, 2002), Figure 7(b) and Cassandra4x4 (Cassandra et al., 1994), Figure7(a) both have aliasing squares and similar average steps to goal (ϕ_m) and density (δ_m) values. However, Cassandra4x4 is much easier to solve than E2, because the aliasing squares of Cassandra4x4 do not affect the optimal strategy (shown with pointers). This is reflected in their widely different complexity values of $\psi_m = 251$ for E2 and $\psi_m = 1$ for Cassandra4x4.

(a)   (b)

Figure 7: (a) Cassandra4x4, $\delta = 0.38$, $\phi = 2.27$, $\psi = 1$; (b) E2, $\delta = 0.25$, $\phi = 2.33$, $\psi = 251.2$

(Lanzi, 1997) noticed that disposition of aliasing cells play a significant role in maze complexity. We assume that for most LCS agents there are two major factors that have a significant influence on the learning

process: minimal distance to food, d, and correct direction to food, or right action, a. Let d_1 and d_2 be minimal distance to food from an aliasing cell 1 and aliasing cell 2 respectively, and a_1 and a_2 be the optimal actions for the cells. There are four different situations for that case:

- both distance and direction are the same ($d_1 = d_2$, $a_1 = a_2$); the squares are *pseudo-aliasing*;

- distance is different but direction is the same ($d_1 \neq d_2$, $a_1 = a_2$); *type I* aliasing squares;

- distance is different and direction is different ($d_1 \neq d_2$, $a_1 \neq a_2$); *type II* aliasing squares;

- distance is the same but direction is different ($d_1 = d_2$, $a_1 \neq a_2$); aliasing *type III*.

Thus, there are three types of genuine aliasing squares and one type of pseudo-aliasing conditions. Woods2 is an example of a maze with pseudo-aliasing cells. It can be seen from Figure 8(a) that for Littman57 (Littman, 1995) the aliasing cells marked with 1, 2 and 3 have the same direction to food (aliasing type I). Figure 8(b) shows MazeF4 (Stolzmann, 2000) with aliasing squares type II marked with 1. Both squares have different distances to food as well as different directions. Woods101 (McCallum, 1993), Figure 10(a) is an example of a maze with type III aliasing squares.

(a)   (b)

Figure 8: (a) Littman57, aliasing maze type I; (b) MazeF4, aliasing maze type II

4.2.2 The Influence of aliasing type on maze complexity

Each aliasing type will produce distinctive kinds of noise in the agent's reward function and understanding the internal structure of the noise may help us to develop a mechanism for improving the learning of the agent. The obtained results show that the mazes with a large value of ψ_m ($\psi_m > 150$) all have type III aliasing squares (see Figure 9). The majority of mazes that include aliasing type II squares as the highest aliasing have $10 \leq \psi_m \leq 150$. Mazes that include only aliasing type I produce a $\psi_m < 10$. Each maze can then be categorized by the type of aliasing cells it includes. For mazes that have *combined* aliasing (more than one aliasing type), we define the aliasing group a maze belongs to by the highest aliasing type it contains. Thus, aliasing mazes type III may be considered as the most difficult group of aliasing mazes, mazes type II are of medium complexity and those type I are the easiest.

Table 4 presents distribution of the collected mazes and publications by size and aliasing types. It can be seen that the majority of mazes used in research is non-aliasing, and most papers use small or medium sized mazes.

4.2.3 Distance to food with aliasing squares

A learning agent that is able to perceive only the surrounding squares may not be able to achieve the average distance to food value for the maze, even though it is capable to solve the maze. This is because when the
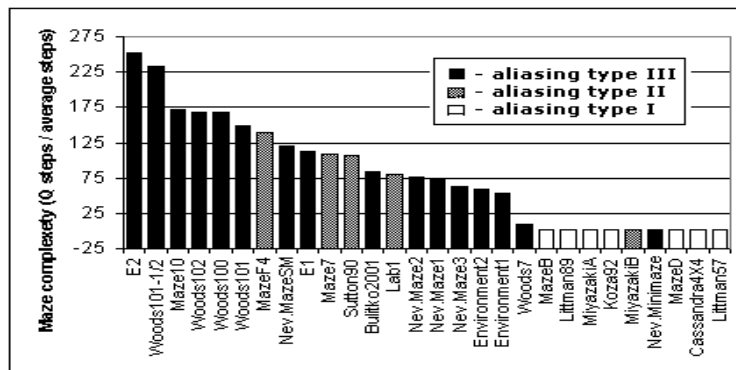
Figure 9: Maze complexity chart.

Table 4: Distribution of mazes and publications by size and aliasing types (figure in brackets is the number of publications).

	Non-alias	Alias I	Alias II	Alias III	Total
Small	13 (28)	1 (2)	2 (6)	2 (16)	18 (51)
Medium	7 (30)	6 (9)	2 (7)	5 (12)	20 (57)
Large	1 (10)	0	1 (1)	3 (8)	5 (19)
Total	21 (42)	7 (8)	5 (13)	10 (18)	43 (63)

initial position happens to be an aliasing square it cannot ever be certain of making the correct decision, as it has no other information to allow it to disambiguate its location. Hence, it is unfair to compare the achieved average steps to food with the optimal ϕ_m, as this level of performance may be unreachable. Instead, to find the optimal performance level for an agent with limited perceptive power, we calculate the *alias-expected* average steps-to-food, ϕ'_m.

Let $A^{opt}(sq)$ be the set of optimal actions for a square in a maze, $s(sq)$ be the state perceived in the square, Sq be the set of aliasing squares in the aliasing state, and $A^{opt*}(s)$ be the set of optimal actions for the aliasing state. Let $sq \in Sq$ be the position of the agent, and $d(sq)$ be the expected optimal distance to food from the square for the agent with limited perceptive power. Then $d(sq)$ is calculated as follows:

$d(sq) = 0$, if sq contains the food object;

$d(sq) = d_{min}(sq)$ for all distinct squares, for distinguished aliasing squares and for aliasing squares where the set of optimal actions is the same for each square in the aliasing group Sq (aliasing type I and pseudo-aliasing), where d_{min} is the actual minimal distance to food from the square;

$d(sq) = \bar{d}(Sq'|A^{opt*}(s), sq) + 1$, for all other aliasing squares, where Sq' is the set of next possible squares, when the agent performs each action from the set of actions $A^{opt*}(s)$ being in the square sq, and $\bar{d}$ is the average distance to food from all squares in Sq'. Recursive calculations are applied where appropriate.

When the expected optimal distance to food from every square available for the agent is defined, the expected optimal performance on the maze is calculated as the average distance to food in the obtained matrix. Based on a comparison of the calculations we have made and the published figures, we believe that ϕ' has been the measure most commonly used in maze research (Lanzi, 1997; Lanzi and Wilson, 1999;

Stolzmann, 2000) as the optimal performance level (although it is not often explicitly stated).

4.2.4 Further aliasing metrics

Solving an aliasing non-Markov maze implies bringing it to the condition where it becomes Markovian and hence is predictable for the agent. Thus, it is the agent's structure and abilities that make an aliasing maze Markov or non-Markov, while dynamic mazes are completely agent-independent in their non-Markov properties.

Different learning systems may have different attributes that influence complexity. For example, agents that belongs to the class of predictive modelling systems, like Anticipatory Classifier Systems (ACS) (Stolzmann, 2000), predict not only reward, but also the next environment state s'. Aliasing can thus be more complex and a wider classification may be suitable:

- $d_1 = d_2$, $a_1 = a_2$, $s'_1 = s'_2$ — pseudo-aliasing, and
  $d_1 = d_2$, $a_1 = a_2$, $s'_1 \neq s'_2$ — pseudo-aliasing, predictive mismatch

- $d_1 \neq d_2$, $a_1 = a_2$, $s'_1 = s'_2$ — type I, and
  $d_1 \neq d_2$, $a_1 = a_2$, $s'_1 \neq s'_2$ — type I, predictive mismatch

- $d_1 \neq d_2$, $a_1 \neq a_2$, $s'_1 = s'_2$ — type II, and
  $d_1 \neq d_2$, $a_1 \neq a_2$, $s'_1 \neq s'_2$ — type II, predictive mismatch

- $d_1 = d_2$, $a_1 \neq a_2$, $s'_1 = s'_2$ — type III, and
  $d_1 = d_2$, $a_1 \neq a_2$, $s'_1 \neq s'_2$ — type III, predictive mismatch

In addition, some aliasing mazes may have *aliasing chains*, like Woods102 (Cliff and Ross, 1994) (Figure 10(b)) with adjacent aliasing squares 1 and 2. The chains may be composed of different aliasing states, or, on the contrary, of the same aliasing states (e.g. E2). Other mazes may have *communicating aliasing cells*, i.e. two aliasing cells bordering on the same neighbour cell (for example, Woods101 (Figure 10(a)). Mazes with chains or communicating aliasing cells may present a task of increased complexity for some types of predictive modelling agents.

(a)   (b)

Figure 10: (a)Woods101, aliasing maze type III, communicating aliasing cells; (b)Woods102 (fragment), aliasing chains

4.3 Generalization and Uncertainty

Generalization. In a maze task at each time step the agent receives a signal that comprises the information about current environment state, usually a coded reflection of the surrounding cells consisting of 'zero' and 'one' symbols. Generalization is reducing the number of significant bits used to represent the environment

situation. The process groups similar types of states together in a less specialized state based upon common attributes and substitutes 'zero' and 'one' with a 'don't care' symbol (usually '#' symbol) as shown in Figure 11. The goal of generalization is to extend the range of the states that can be represented by a smaller population without being overgeneralized. The main question is how the right generalization can be differentiated from overgeneralization.

$$10101010111111100 \quad = \quad 1\#10101\#11111100$$
$$11101011111111100$$

Figure 11: Generalization.

Any generalization process applied to a maze introduces aliasing. If the generalized states have the same distance and directions to food (i.e. if they fall into the pseudo-aliasing category), the generalization is correct and beneficial. Generalization leading to the aliasing type I (the same directions, different distance) also can be beneficial. However, some disturbance to the learning process should be expected because of continuous changes in the reward function. Any generalized state that contains aliasing type II or III is overgeneralized, because the squares concealed in the state always demand completely different actions. Table 4.3 illustrates this categorization.

Table 5: Results of generalization: '+' means the generalization is beneficial, '-' means the generalization is detrimental.

pseudo-aliasing	$d_1 = d_2, a_1 = a_2$	$++$
aliasing type I	$d_1 \neq d_2, a_1 = a_2$	$+-$
aliasing type II	$d_1 \neq d_2, a_1 \neq a_2$	$--$
aliasing type III	$d_1 = d_2, a_1 \neq a_2$	$--$

Noise. Noise is a disturbance of a random nature in the agent's information system, bringing an uncertainty either to its actions or to the environment signals it receives. If there is a *detector noise* than the perceived state may be different to the actual state, i.e. s_{per} is a probabilistic function of the original environment state s: $s_{per} = f(s)$. Thus, each environment state would correspond to a set of perceived environment states:

$$s \Rightarrow s_{per1}, s_{per2}, \ldots s_{pern}$$

The size of the set depends on the noise function. The maximal size of the set is limited to the number of states the detector is able to perceive. For example, for a detector represented as a binary string it would be $N_{per} \leq 2^n$, where N_{per} is the maximal number of the perceived states corresponding to a single environment state and n is the number of bits in the detector.

Detector noise can increase the number of states perceived by the agent significantly. The problem of the increase of the number of the perceived states could be addressed with an appropriate generalization technique, so that $S_i^{per} \Rightarrow s_i, \forall i$, where S_i^{per} is the set of perceived states for the environment state s_i. However, the feasibility of such generalization depends on the sets of perceived states. If for each two states s_i and s_j in a non-aliasing maze the sets of perceived states do not intersect, the noise introduces pseudo-aliasing and the successful generalization can be achieved. Otherwise, if the sets of perceived states are

intersecting, the noise function introduces aliasing and the performance of the learning agent in the maze is considerably affected.

Effector noise means that the action conducted may be different to the action selected, i.e. a_{cond} of the agent is a probabilistic function of the original action a: $a_{cond} = f(a)$. Thus, for each action-state pair (s^{t-1}, a) in the environment, there will be a set of possible next environment states $\{s_1^t, s_2^t, \ldots, s_n^t\}$. However, the maximal size of the set cannot be greater than the number of actions available for the agent.

Effector noise always introduces aliasing, hence, the problem cannot be addressed with a generalization technique. Although, unlike with detector noise, the total number of perceived states in the case of effector noise is the same as the number of actual environment states in the maze, which significantly reduces noise impact on the memory requirements, compare to detector noise. Detailed analysis of generalization and noise problems in maze environments can be found in (Bagnall and Zatuchna, 2005a).

4.4 A Summary Maze Complexity Criteria

The characteristics of mazes that determine the complexity of the learning task fall into two classes: those that are independent of the learning algorithm, such as number of squares and density of obstacles; and those that are an artifact of the agents ability to correctly detect its current state, such as the number and type of aliasing cells. Table 6 lists the main variables we propose to measure maze complexity.

Table 6: Maze Complexity Measures

Name	Definition
Size, s	Total number of cells in a maze
Density, δ	Proportion of obstacles to squares
Distance, ϕ	Average distance to goal square
Optimal performance, ϕ'	Alias-expected average distance to goal square
Q-Steps, ϕ^Q	Average steps taken by trained Q-learning
Learning Complexity, ψ	ϕ^Q/ϕ
Aliasing Type	The highest type of aliasing square present
Valency, ν	Number of aliasing squares in an aliasing state
Aliasing Number	Number of aliasing squares in a maze

4.5 Maze Assessment Software System (MASS)

The Maze Assessment Software System (MASS) is software capable of analyzing maze domains by: width; height; average steps to goal; max steps to goal; density; number of pseudo-aliasing and aliasing states and squares; average Q-learning steps, types and location of aliasing squares. MASS also produces the following outputs: transition matrix; step-to-food map; Q-learning coefficient map; Q-learning step map. Further information about MASS software can be found at (Zatuchna and Bagnall, 2005b).

5   Results

For comparison purposes, in Section 5.1 we evaluate the performance of AgentP in a way consistent with that presented in the LCS literature (Stolzmann, 2000; Metivier and Lattaud, 2002; Lanzi, 1997; Lanzi and Wilson, 1999; Cliff and Ross, 1994; Lanzi, 1998). The method of presenting results in these papers is to plot a graph of the average number of steps to food over 50 trials against number of exploit trials. This gives a good indication of performance, but the lack of actual data makes comparison difficult. Hence in Section 5.2 we describe performance metrics to measure the following: correctness (whether the agent has learnt the optimal policy); convergence (steps to a stable policy); and memory (number of rules required for the converged policy). We then present the average and standard deviation of these performance metrics for AgentP on the mazes listed in Section 5.1.

5.1   Graphical Results

XCSM and XCSMH (Lanzi, 1997; Lanzi and Wilson, 1999; Cliff and Ross, 1994; Lanzi, 1998) have been applied to aliasing mazes Woods101, Woods101.5, Woods102, Maze7 and Maze10. ACS (Stolzmann, 2000; Metivier and Lattaud, 2002) has been used with aliasing mazes Woods100, MazeF4 and E1. Table 7 gives the complexity measures for these 8 mazes.

Table 7: Complexity of aliasing mazes used in the LCS literature

	Woods 100	Woods 101	Woods 101-1/2	Woods 102	Maze F4	E1	Maze7	Maze10
Aliasing	III	III	III	III	II	III	II	III
Size	7	11	22	28	11	45	10	19
Density	0.59	0.56	0.64	0.54	0.65	0.31	0.6	0.54
ϕ	2	2.7	2.8	2.77	4.3	2.45	4.11	4.56
$\phi\prime$	2.33	2.9	3.1	3.31	4.5	3.07	4.33	5.17
Q	166	149	251	167	47	167	82	171
Non-A	4	6	12	12	8	24	7	5
Pseudo	0	2	4	8	0	0	0	0
Type I	0	0	0	0	0	0	0	6
Type II	0	0	0	0	2	0	2	5
Type III	2	2	4	6	0	16	0	2

In all of these papers except (Metivier and Lattaud, 2002) the results are presented as a graph without any quantitative indication of how well the LCS is solving the maze. In (Metivier and Lattaud, 2002) it is stated that ACS *"converges to a performance around 3.3"*. In all papers it is claimed that the LCS reaches an optimal or near optimal solution.

We have estimated the best performance statistics for versions of XCS and ACS and presented them alongside those for AgentP in Table 8.

These results are at least as good as the best results reported in (Lanzi, 1997; Lanzi and Wilson, 1999; Cliff and Ross, 1994; Lanzi, 1998), and indicate that AgentP may be performing better on maze E1. Firstly,

Table 8: Table of performance of different LCS agents on the aliasing mazes. The numbers in brackets represent the population size. Results from [1] (Metivier and Lattaud, 2002). [2] (Cliff and Ross, 1994). [3] (Bull and Hurst, 2001). [4] (Lanzi, 1997). [5] (Lanzi and Wilson, 1999). [6] (Stolzmann, 2000). [7] (Lanzi, 1998).

Maze \ LCS	ϕ_m / ϕ'_m	ZCSM (N)	XCSM, XCSMH (N)	ACS ($N_{\neq}$)	AgentP ($N_{\neq}$)
Woods100	2 / 2.33	—	—	2.3(60)[1]	2.3(10)
Woods101	2.7 / 2.9	5(400)[2], 2.9(400)[3]	3(800)[4], 2.9(800)[5]	—	2.9(27)
Woods101.5	2.8 / 3.1	—	3.1(2800)[5]	—	3.1(42)
Woods102	2.77 / 3.31	9(400)[2]	5(2400)[4], 3.3(6000)[5]	—	3.3(82)
MazeF4	4.3 / 4.5	—	—	4.5[6]	4.5(25)
Maze7	4.11 / 4.33	—	10(1600)[4], 4.3(1600)[5]	—	4.3(21)
E1	2.45 / 2.84	—	—	3.3(2800)[1]	2.84 (240)

AgentP learns faster that the other LCS. For example, AgentP needs 22 trials (440 steps) on average to reach the optimal performance on Woods100, while it takes 2000 steps to reach 100% knowledge for the maze by ACS (Metivier and Lattaud, 2002). Similar comparison can be made for MazeF4, with 22 trials on average, while ACS (Stolzmann, 2000) which has been reported to need more than 500 trials. Secondly, AgentP needs fewer classifiers to solve a maze than XCS or ACS do. For example, ACS reached a sub-optimal solution on E1 with 2800 rules, but AgentP solved the maze optimally with only 240 rules. Once AgentP reaches the optimal, the performance level remains stable and the number of classifiers stays constant. Figure 12 presents the summary diagrams for the performance of Self-Adjusting and Gradual AgentP on the mazes. We can judge from these summary measures and graphs that AgentP reaches the optimal performance for these mazes, achieving the same results as the other LCS agents in terms of steps to food, but outperforming them in terms of speed, computation resources and stability.
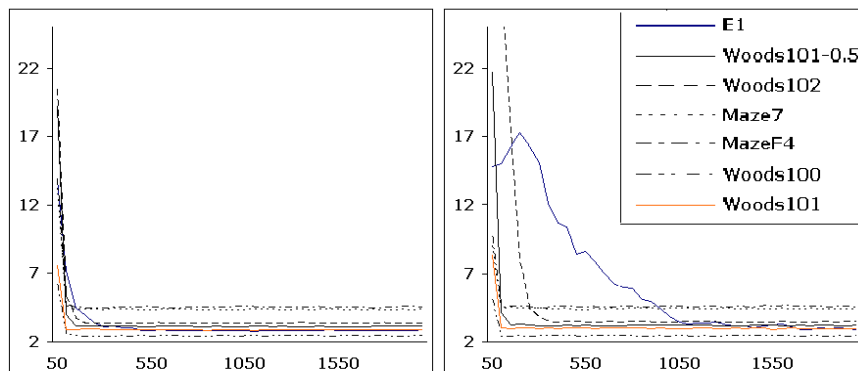

Figure 12: Average steps to food for Self-Adjusting (left) and Gradual (right) AgentP on 7 LCS mazes. Results are averaged over 50 separate runs.

Seven other aliasing mazes have been used in the RL literature, although not with LCS. These are

Table 9: Complexity of aliasing mazes used in the RL literature

	Littman57	Littman89	MazeB	MazeD	MiyazakiA	MiyazakiB
Aliasing	I	I	I	I	I	II
Size	52	63	64	64	64	64
Density	0.59	0.53	0.48	0.5	0.4	0.44
ϕ	3.71	3.77	3.48	2.71	3.03	3.3
$\phi\prime$	3.71	3.77	3.5	2.75	3.05	3.33
Q	154	61	1.26	1.03	69	1.03
Non-A	6	16	19	22	21	21
Pseudo	0	0	0	0	0	0
Type I	8	6	6	2	8	4
Type II	0	0	0	0	0	2
Type III	0	0	0	0	0	0

Littman57, Littman89 (Littman, 1995), MazeB, MazeD (Arai and Sycara, 2001), MiyazakiA and Miyaza-kiB (Miyazaki and Kobayashi, 1999). The complexity measures are summarised in Table 9. Unfortunately, the majority of non-LCS papers on mazes do not provide any comparable information on performance. The performance of Self-Adjusting and Gradual AgentP on these mazes is shown in Figure 13. Both Self-Adjusting and Gradual types converge to the optimal strategy in all mazes in less than 200 exploit trials in all mazes except for Littman57.
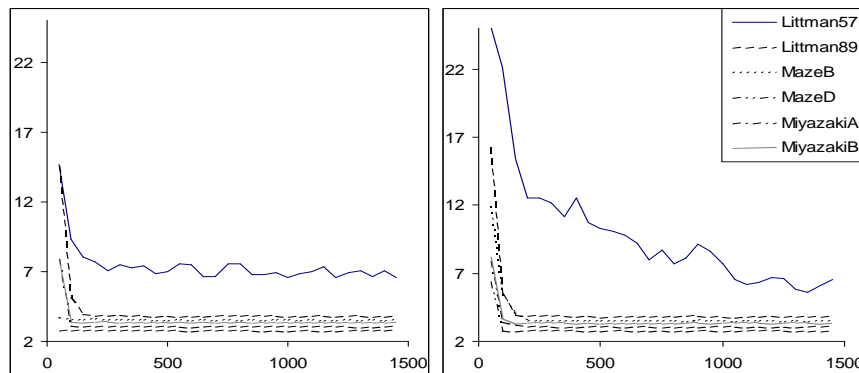

Figure 13: Average steps to food for Self-Adjusting (left) and Gradual (right) AgentP on 6 RL mazes. Results are averaged over 50 separate runs.

Both set of results support the hypothesis that AgentP can learn a wide class of mazes in time less than reported for other LCS (Zatuchna and Bagnall, 2005a). However, the results as presented do not provide enough information to test whether there is significant deviation from optimal, nor to test the hypothesis that one algorithm is better than another. Hence we present more comprehensive results in Section 5.2 in a format that will allow other researchers to compare their results with ours.

5.2 Quantified Results

To properly measure how well an agent is performing on a maze we measure the following statistics for correctness, speed to convergence and memory over repeated learning runs.

Correctness: The 50 step moving average is not necessarily an accurate measure of how close the agent is to the optimal strategy. Since most of the mazes have a relatively small number of states, it is possible to measure exactly how many steps to food an agent needs from every possible starting position (freezing all adaptive processes beforehand) and hence form the average, which is directly comparable to ϕ'. This statistic, c, is measured every 10 exploit trials. This gives us a means of testing precisely whether the agent finds the optimal policy and, potentially, to test whether there is a significant difference between the suboptimal performance of different learning algorithms.

Speed to Convergence: A further measure of quality for mazes is the speed to which the agent converges to a stable policy. We measure this by periodically freezing all learning mechanisms and measuring the correctness statistic, c. If the correctness measure changes little over a long period, we assume the learning algorithm has converged to a strategy. We define convergence as deviation of less than 10% percent over 500 trials.

Memory: The total number of rules provides an indication of memory required that is sufficient at this moment. However, a more complex memory definition would be required if one wished to compare the memory requirements of different learning algorithms, as rules for different LCS require different amounts of memory.
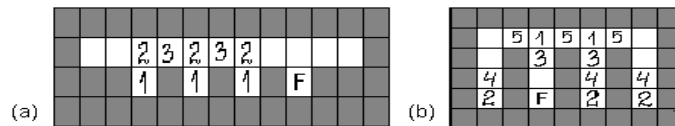

Figure 14: (a) Littman57; (b) Maze10. Aliasing squares marked with digits.

An experiment involves measuring the correctness, convergence and memory for a learning algorithm starting with no prior knowledge. For any maze we repeat an experiment 50 times. Of the 14 mazes considered in Section 5.1, both the Self-Adjusting and Gradual AgentP reached the optimal policy ($c = \phi'$) on every run on all mazes except Littman57 and Maze10 (shown in Figure 14). On Littman57 (optimal 3.71 steps to food) Self-Adjusting found the optimal on 35 runs and Gradual on 48 runs. Self-Adjusting AgentP generally found good solutions whereas Gradual AgentP tended to either find the optimal or converge to a very poor solution. This is reflected by the correctness statistics given in Table 10. The average for Gradual AgentP is lower, but the variance is much higher. This is in line with expectations: Self-Adjusting is more flexible and more likely to find a good, but suboptimal, solution. Gradual is more likely to solve the problem, but if it goes wrong early in the experiment it is unable to correct the mistake. Gradual performed much better on Maze10, finding the optimal on 41 occasions and achieving an average comparable to the results estimated from the graph in (Lanzi, 1998).

Table 11 represents speed to convergence results for AgentP on 14 collected aliasing mazes. Table 12 shows the average number of rules for both agent types at the end of the run, with the standard deviation in

Table 10: Mean and standard deviation correctness statistics for 50 runs on Littman57 and Maze10 for Self-Adjusting and Gradual AgentP. The first result in each cell is for Self-Adjusting AgentP

	ϕ'	Optimal Runs	Mean c	Standard Deviation c
Littman57	3.71	35/48	5.24/4.82	2.80/5.41
Maze10	5.11	0/41	31.71/7.87	5.97/7.43

brackets if indeed there was any deviation. It is notable that Self-Adjusting has a significantly smaller final rule set than Gradual on Maze10, indicating that it is failing to adequately identify aliasing squares that require differentiation.

Table 11: Trials to convergence over 50 runs for Self-Adjusting and Gradual AgentP on 14 mazes. Significantly different mean times are shown in bold

	Self-Adjusting		Gradual				Self-Adjusting		Gradual	
Mazes	Av.	SD	Av.	SD		Mazes	Av.	SD	Av.	SD
Woods100	21	13.09	17	6.88		E1	277	107.56	**621**	253.75
Woods101	25	7.07	23	8.14		Littman57	100	23.23	582	646.83
Woods101.5	48	16.07	**67**	20.45		Littman89	57	21.63	**100**	30.67
Woods102	48	16.07	**234**	46.25		MazeB	66	27.11	**113**	28.91
Maze7	**25**	12.81	18	5.07		MazeD	46	22.12	37	13.67
Maze10	204	317.85	537	421.37		MiyazakiA	57	18.32	**145**	41.56
F4	22	9.38	19	7.40		MiyazakiB	45	13.44	**85**	28.66

Table 12: Average number of rules $N_{\neq}$ for Self-Adjusting and Gradual AgentP. The figure in brackets is the standard deviation, incuded for mazes where the agent did not always finish with the same number of mazes

	Self-Adjusting	Gradual		Self-Adjusting	Gradual
Woods100	10	10	E1	240	240
Woods101	27	27	Littman57	41	40.52 (1.83)
Woods101.5	42	42	Littman89	77	77
Woods102	82	82	MazeB	102	102
Maze7	21	21	MazeD	96	96
Maze10	35.1 (1.37)	46.22 (2.30)	MiyazakiA	150	150
F4	25	25	MitazakiB	140	140

Hence we conclude that AgentP can learn at least as well as other LCS in less time and with less memory. The definition of three explicit performance metrics will allow for a more accurate comparison of different learning mechanisms in the future. However, the results in this Section also demonstrate that of the mazes used in the literature, only Maze10 and Littman57 present a challenge to either forms of AgentP. We believe that future comparisons will be more meaningful if performance is evaluated on mazes with a wider range of complexity. Hence in Section 5.3 we evaluate AgentP on a large set of new mazes.

5.3   New Mazes

Most of the mazes used to date in the literature have been constructed to illustrate a particular research point. Our interest lies more in being able to compare the ability of algorithms to solve mazes on classes of problem. For example, we may wish to test the hypothesis that ACS learns a better strategy on the class of Type I aliasing problem than XCS (note that this is a fabricated example not the authors' opinion). To test such hypothesis a larger set of test mazes is required. We have generated several hundred mazes randomly by conducting a modified random walk through a maze of only wall squares to leave a trail of empty cells. We then located the food randomly in a non-wall cell. This ensures that the maze is solvable from all starting positions. The mazes thus generated were classified with the MASS software described in Section 4. We began by running both forms of AgentP on non-aliasing mazes and verified that both versions could solve the simplest form of maze in all cases.

Figure 15: Toroidal mazes (a) AliasIIMaze20; (b) AliasIIIMaze20

Three sets of 10x10 mazes were selected, twenty each of type I, type II and type III aliasing maze. We name these mazes AliasIMaze1, ...,AliasIMaze20, AliasIIMaze1,...AliasIIMaze20 and AliasIIIMaze1, ... AliasIIIMaze20. Examples of the mazes are shown in Figure 15. A full description of their characteristics and a complete set of correctness, convergence and memory statistics for both versions of AgentP are given at (Zatuchna and Bagnall, 2005b). Table 13 gives the number of mazes solved every time (for 50 runs) by both types of agent. It demonstrates that, firstly, both learning algorithms are able to solve mazes with just Type I aliasing squares every time and secondly, that they are both solving approximately 75-80% of mazes with Type II and Type III aliasing squares.

Table 13: Number of mazes solved by Self-Adjusting and Gradual AgentP

Maze Type	Self-Adjusting	Gradual
Alias I	20 / 100%	20 / 100%
Alias II	16 / 80%	17 / 85%
Alias III	15 / 75%	15 / 75%

The number of mazes for which AgentP is not able to find an optimal policy is the same for the two learning modes. Some mazes that represent a task of higher difficulty for Self-Adjusting AgentP are easily solved by Gradual, and vice versa. Table 14 shows the correctness statistics for the 10 mazes that at least one of the agents was unable to solve every time.

Table 14: Correctness for mazes not solved on every run by either Self-Adjusting or Gradual. A blank square indicates that particular agent solved that maze.

Maze	ϕ'	Self-Adjusting		Gradual	
		Average	St Dev	Average	St Dev
AliasIIMaze12	2.21	2.26	0.38		
AliasIIMaze13	3.30	9.09	6.33	6.55	6.22
AliasIIMaze18	3.91	5.26	4.51	4.35	1.99
AliasIIMaze19	3.49			3.75	1.18
AliasIIMaze20	7.31	12.05	8.23		
AliasIIIMaze13	3.32	4.99	3.27	3.73	2.00
AliasIIIMaze16	3.80			4.39	2.07
AliasIIIMaze17	3.64	4.87	3.49		
AliasIIIMaze18	4.35			4.52	0.52
AliasIIIMaze19	3.62	5.67	2.15	6.02	4.34
AliasIIIMaze20	4.28	4.95	1.70	8.90	5.94

Figure 16 shows the box plot for the steps to convergence by maze type. This data illustrates the following: firstly, Self-Adjusting converges to a stable strategy faster on average for all maze types. Secondly, both converge faster on Alias Type I mazes than on other types. Thirdly, whereas for Self-Adjusting there is little difference between Type II and Type III, Gradual AgentP takes longer on Type III mazes. Finally, there is greater variation in convergence for Gradual AgentP on all type of mazes.
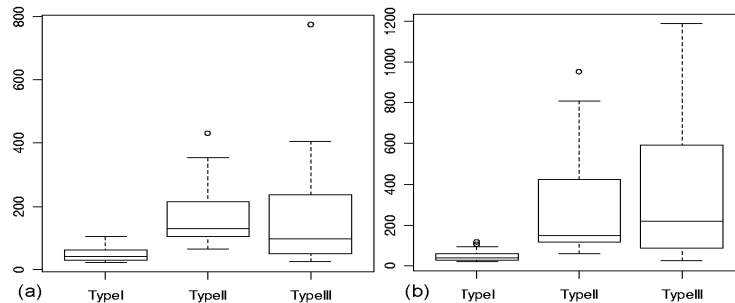

Figure 16: Convergence distributions over Type I, Type II and Type III aliasing mazes for (a) Self-Adjusting (b) Gradual.

The difference between average types I and II aliasing mazes is not significant because of the effect of the physical factors of the mazes. To demonstrate that physical complexity is also a factor in convergence, Figure 17 shows the scatter plot of steps to food against trials to convergence. This graph also demonstrates the longer time to convergence for Gradual AgentP.

The number of rules for Gradual and Self-Adjusting follow a similar pattern to the trials to convergence results, although there is much less difference between the algorithms. Table 15 shows the average final number of rules for each maze class. Figure 18 shows how the size and complexity of the maze, quantified in the steps a trained Q-learning agent requires, correlates well with the number of rules both forms of AgentP require.
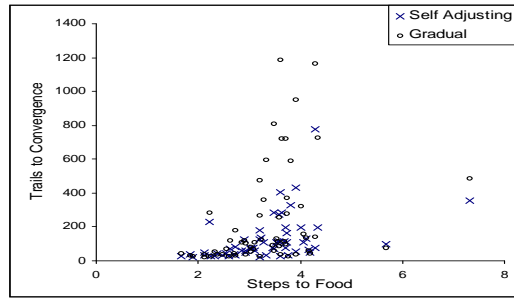
Figure 17: Steps to food plotted against average trials to convergence for AgentP.
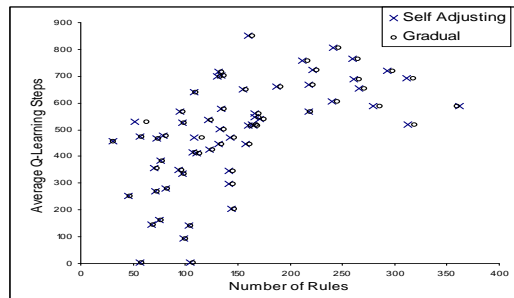

Figure 18: Q-learning steps plotted against average number of rules for AgentP.

Table 15: Average number of rules of Self-Adjusting and Gradual AgentP for three maze classes

Maze Type	Self-Adjusting	Gradual
Alias I	88	89
Alias II	177	181
Alias III	181	185

As an initial step to examining the limits of the current agent, we generated several larger (up to 30x30) mazes (also available from (Zatuchna and Bagnall, 2005b)). The basic results for six of these mazes are shown in Table 16.

Table 16: Average steps to food for six large mazes

Maze name	Optimal	Self-Adjusting	Gradual
LargeAliasIMazeI	7.85	7.85	7.85
LargeAliasIIMazeI	13.02	13.25	13.02
LargeAliasIIMaze2	12.74	34.84	31.03
LargeAliasIIIMazeI	6.72	17.13	6.72
LargeAliasIIIMaze2	5.64	5.64	5.64
LargeAliasIIIMaze3	15.66	34.24	19.42

6 Conclusions

In this paper we introduce a new LCS, AgentP, that is based on the psychological model of Associative Perception Learning, and evaluate it on a variety of aliasing mazes. AgentP learns through a process of imprinting and organisation of images of the environment. AgentP differs from other LCS in several key ways:

- Like ACS, AgentP employs a state/action/state rule structure and like ZCSM, XCSM and XCSMH it employs a memory structure. Unlike any previously proposed LCS, it uses these in conjunction.

- Like ACS, AgentP attempts to detect aliasing states and correct rules that incorrectly predicted the result state from a given initial state/action pair. Unlike ACS, AgentP also performs a backward correction: it detects aliasing in the initial state and adjusts rules that predicted the observed result state, but incorrectly matched the initial state.

- AgentP uses a distance based reward distribution mechanism, that allow to distribute the reward evenly, however large the size of maze is.

- AgentP does not attempt to learn generalizations of states.

AgentP can solve the majority of the aliasing mazes used in the literature very quickly. It learns at least as good a strategy as other LCS in fewer trials using less memory. On some mazes, such as E1, it seems to find a better policy than other LCS. However, a full comparison between alternative LCS is made difficult by the following: firstly, there has been a tendency for researchers using different types of algorithm to use largely non overlapping sets of mazes. Secondly, results have been presented in graphical form without numerical performance metrics. Unfortunately, running new experiments on mazes using different learning agents to replicate and specify the empirical results is not possible because those are not available for public use. Those LCS, which have been widely reproduced and are easily available, such as XCS, have not been designed for the aliasing problem and are not able to cope with it.

We have attempted to rectify the first problem by collating a large number of mazes used in research and by generating new random mazes. In addition, we have defined metrics to estimate the complexity of a maze for a learning algorithm with limited perception. Software to generate random mazes and measure the complexity characteristics, in addition to text and bitmap versions of all mazes used in this paper, is available from (Zatuchna and Bagnall, 2005b).

We have addressed the second issue by defining metrics to estimate the performance of an agent in terms of correctness (a measure of the quality of the agent's policy), convergence (an estimate of the time taken to find a stable policy) and memory (the number of rules required to represent a policy). We ran AgentP repeatedly on each maze, providing a mean and standard deviation of each performance statistic. We hope this approach will make it easier to reproduce our results and allow our results to be compared with those of other learning algorithms.

Both forms of AgentP are always able to solve 10x10 aliasing Type I mazes, and can solve approximately 75% of aliasing Type II and Type III mazes. Gradual AgentP takes longer to converge than Self-Adjusting

AgentP, but also it performs better on the previously published mazes and on some of the new mazes. However, on some of the new mazes Self-Adjusting AgentP finds a better policy than Gradual AgentP. This indicates that Gradual AgentP is at times discounting useful information because it cannot determine its meaning with certainty. This experimental evidence suggests that hybridizing a Gradual agent with a Self-Adjusting agent will lead to improved performance.

At the next stage of the research we are going to examine the limits of the present design of AgentP by running it on a set of larger maze environments. We hope the analysis of the test results will also help us to understand better what makes some maze problems hard for AgentP. Another primary direction for future research is development of a post-learning generalization mechanism to allow AgentP to extract the knowledge obtained in one part of a maze environment into a more compact representation and then use it in other areas of the same environment or different, but similar maze environments.

References

Arai, S. and Sycara,K. (2001) Credit assignment method for learning effective stochastic policies. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 815–822, Morgan.

Bagnall, A. J. and Zatuchna, Z. V. (2005) On the classification of maze problems.In Bull, L. and Kovacs, T., editors, *Foundations of Learning Classifier Systems*, pages 307–316, Springer.

Browne, W. and Scott, D. (2005) An abstraction algorithm for genetics-based reinforcement learning. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1875–1882.

Bull, L. and Hurst, J. (2002) ZCS: Theory and Practice, Tech. Report at Learning Classifier Systems Group, UWE.

Bull, L. (2002) Lookahead and latent learning in ZCS. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 897–904, Morgan.

Bull, L. and Hurst, J. (2002) ZCS redux. *Evol. Comput.*, 10(2):185–205.

Bull, L. and Hurst, J. (2003) A neural Learning Classifier System with self-adaptive constructivism. Tech. Report, University of the West of England.

Butz, M., Goldberg, D. E. and Stolzmann, W. (1999) New challenges for an ACS: hard problems and possible solutions. Techn. Report 99019, University of Illinois at Urbana-Champaign.

Cassandra, A. R., Kaelbling, L. P., and Littman, M. L. (1994) Acting optimally in partially observable stochastic domains. *Proceedings of Conference on Artificial Intelligence*, pages 1023–1028.

Cliff, D. and Ross, S. (2004) Adding Temporary Memory to ZCS. *Adaptive Behavior*, 3(2):101–150.

Gerard, P. and Sigaud, O. (2001) YACS: Combining dynamic programming with generalization in Classifier Systems. In *Advances in Classifier Systems*, pages 52–69, Springer–Verlag.

Holland, J. H. and Reitman, J. S. (1978) Cognitive systems based on adaptive algorithms. *Pattern-directed Inference Systems*. New-York: Academic Press.

Lanzi, P. L. (1997) Solving problems in partially observable environments with Classifier Systems. Tech. Report 97.45, Politecnico di Milano.

Lanzi, P. L. (1998) An analysis of the memory mechanism of XCSM. *Proceedings of the Annual Conference on Genetic Programming*, pages 643–651, Morgan.

Lanzi, P. L. and Wilson, S. W. (1999) Optimal Classifier System performance in non-Markov environments. Techn. Report 99.36, Politecnico di Milano.

Littman, M. L. (1992) An optimization-based categorization of reinforcement learning environments. *Proceedings of the International Conference on Simulation of Adaptive Behavior*, MIT Press.

Littman, M. L. (1995) Learning policies for partially observable environments: Scaling up. *Proceedings of the International Conference on Machine Learning.*

Lorenz, K. (1935) Der kumpan in der umwelt des vogels. *Journal of Ornithology*, pages 137–215.

McCallum, A. R. (1993) Overcoming incomplete perception with utile distinction memory. *The Proceedings of the Tenth International Machine Learning Conference.*

Métivier, M. and Lattaud, C. (2002) Anticipatory Classifier System using behavioral sequences in non-Markov environments. *Advances in Learning Classifier Systems*, pages 143–162.

Miyazaki, K. and Kobayashi, S. (1999) Proposal for an algorithm to improve a rational policy in POMDPs. *Proc. of International Conference on Systems, Man and Cybernetics*, pages 492–497.

Nevison, C. (1999) Maze lab 1: Event loop programming, Colgate University.

O'Hara, T. and Bull, L. (2005a) A memetic accuracy-based neural Learning Classifier System. *Proceedings of the Congress on Evolutionary Computation*, pages 2040–2045.

Pavlov, I. P. (1927) *Conditioned Reflexes.* London: Oxford University Press.

Pear, J. (2001) *The Science of Learning*, Psychology Press.

Russell, S. and Norvig, P. (1994) *Artificial Intelligence: Modern Approach*, Prentice Hall.

Stolzmann, W. (2000) An introduction to Anticipatory Classifier Systems. *Learning Classifier Systems. From Foundations to Applications*, pages 175–194, Springer-Verlag.

Sutton R. S. and Barto, A. G. (1998) *Reinforcement Learning: An Introduction*, MIT Press.

Thorndike, E. L. (1911) *Animal Intelligence.*, Hafner.

Tolman, E.C. (1932) *Purposive Behaviour in Animals and Men*, New York: Appleton.

Watkins C. J. C. H. and Dayan, P. (1992) Q-learning. *Machine learning*, 8(3):272–292.

Wilson, S. W. (1991) The Animat Path to AI. *Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 15–21, MIT Press.

Wilson, S. W. (1994) ZCS: A zeroth level Classifier System. *Evolutionary Computation*, 2(1):1–18.

Wilson, S. W. (1995) Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175.

Zatuchna Z. V. (2004a) To the Studies on Maze Domains and Maze Domains Classification in the Framework of Learning Classifier Systems Research. Tech. Report CMP-C04-02, University of East Anglia.

Zatuchna Z. V. (2004b) AgentP model: Learning Classifier System with associative perception. In Xin Yao et al., editor, *Proceedings of PPSN conference*, pages 1172–1182, Springer.

Zatuchna Z. V. (2005) *AgentP: A Learning Classifier System with Associative Perception in Maze Environments.* PhD thesis, School of Computing Sciences, UEA.

Zatuchna Z. V. and Bagnall, A. J. (2005a) AgentP classifier system: Self-adjusting vs. Gradual approach. *Proceedings of the Congress on Evolutionary Computation*, pages 1196-1203.

Zatuchna Z. V. and Bagnall, A. J. (2005b) Maze material for AgentP, 2005. http://www.cmp.uea.ac.uk/Research/kdd/projects.php?project=17.