

# Experiments with Reinforcement Learning in Environments with Progressive Difficulty

Michael G. Madden<sup>1</sup> and Tom Howley<sup>1</sup>

**Abstract.** This paper introduces Progressive Reinforcement Learning, which augments standard Q-Learning with a mechanism for transferring experience gained in one problem to new but related problems. In this approach, an agent acquires experience of operating in a simple domain through *experimentation*. It then engages in a period of *introspection*, during which it rationalises the experience gained and formulates symbolic knowledge describing how to behave in that simple domain. When subsequently experimenting in a more complex but related domain, it is guided by this knowledge until it gains direct experience. A test environment with 15 mazes, arranged in order of difficulty, is described. Experiments in this environment demonstrate the benefit of Progressive RL relative to a basic RL approach in which each puzzle is solved from scratch.

## 1 INTRODUCTION

Reinforcement Learning (RL) is sometimes considered a microcosm of all Artificial Intelligence: an intelligent agent is situated in an environment and must learn to operate successfully within it. The agent explores its environment, receiving rewards and punishments, and seeks to maximize its long-term reward. The appeal of RL is that it is applicable in many domains where it is infeasible to specify explicitly how to perform a task, where proficient humans cannot fully articulate how they do what they are doing. Indeed, Dreyfus and Dreyfus [8] claim that a characteristic of expert level skill in a domain is that one operates at a sub-conscious level, so that it is not generally possible for the expert to specify clear rules that govern how he/she is operating. While this claim is often disputed, the benefit of RL in complex situations is clear: even if one cannot say *how* to do a task well, one can say *whether* it is done well, and provide feedback in the form of a positive or negative reward at the completion of a task. However, a shortcoming of basic approaches to RL is that they discover solutions to individual problems, rather than classes of problems.

Section 2 provides a context for this work by introducing a set of maze problems called the *Theseus and the Minotaur* mazes [1]. A key feature of these mazes is that they exhibit progressive difficulty: a human player would generally find the more complex mazes intractable without having gained experience from solving

the less complex ones. Section 3 then describes our Progressive Reinforcement Learning approach, which seeks to augment standard Q-Learning [20] [25] with a mechanism for transferring experience gained in one problem to a new, related, problem. Results of experiments in applying Progressive Reinforcement Learning to the *Theseus and the Minotaur* mazes are presented in Section 4. Then, Section 5 describes related research. Finally, conclusions are drawn and avenues for future research are identified briefly in Section 6.

## 2 AN ENVIRONMENT WITH PROGRESSIVE DIFFICULTY

The *Theseus and the Minotaur* maze puzzles were invented by Robert Abbott for a human audience [1]. In these mazes, the player controls Theseus (grey), who must reach the exit without encountering the Minotaur (black). Theseus has five possible moves: Up, Down, Left, Right and Delay. The Minotaur takes two steps for each step taken by Theseus, following a fixed policy: it tries to move first horizontally and then vertically, provided the move takes it closer to Theseus. Abbott first published a single, paper-based maze, and a software developer named Toby Nelson implemented the maze as a Java applet and used a basic form of genetic algorithm to design other maze layouts. The applet now features 15 mazes of various sizes, arranged in order of increasing complexity. (Complexity is determined by the size of the decision space as well as the number of steps to the solution.) Figure 1 shows some of the mazes.

Because of their progressive complexity, these mazes represent an interesting challenge for RL. To adapt them for use with RL, we have developed a Java program in which the original applet executes, substituting for the web browser in which the applet would ordinarily run. This program allows the learner to control Theseus and to get information from the applet about the state of the maze. Unlike human players, the RL agent does not have a high-level view of a maze: it does not know about walls, nor does it know what actions will lead to positive and negative rewards. In typical RL fashion, the agent must discover all of this through trial and error.

As a baseline, tabular Q-learning [25] was applied to the *Theseus* environment. As formulated, each state corresponds to one particular combination of the coordinates of both Theseus and the Minotaur. Theseus has a choice of 5 actions: North, East, South, West and Delay. A lookup table of Q-values is maintained, e.g. for

---

<sup>1</sup> Department of Information Technology, National University of Ireland, Galway, Ireland. email: michael.madden@nuigalway.ie; hth@eircom.net

Maze 1 (6×6) a table of 6480 entries (5 actions × 1296 states) is required. All Q-values are initialised to 0. After an action is carried out, the Q-value for that state-action pair is updated according to the standard Q-learning equation:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s,a)]$$

where  $r$  is the reward received for the action  $a$  in state  $s$ ,  $\alpha$  is the step-size parameter,  $\gamma$  is the discount factor and  $\max_{a'} Q(s', a')$  is the estimate of the maximum cumulative reinforcement that the agent will receive from the next state  $s'$  onwards. A reward of 1 is given for reaching the exit, a reward of -1 is given for reaching the Minotaur and a small negative reward,  $r_n$ , is given in non-terminal states. To avoid stalemate situations, when the same state and action path is traversed on two consecutive moves, the play is ended and a reward of -1 is given. Exploration is achieved by following an  $\epsilon$ -greedy policy, in which the action with the highest Q-value is chosen with probability  $1-\epsilon$ .

Preliminary experiments were carried out to determine good values for the parameters  $\alpha$ ,  $\gamma$ ,  $\epsilon$  and  $r_n$ . From these, the values found to give the best performance were  $\alpha = 0.3$ ,  $\gamma = 0.9$ ,  $\epsilon = 0.01$  and  $r_n = -0.02$ . Results of the baseline Q-learning experiments are reported in Section 4.

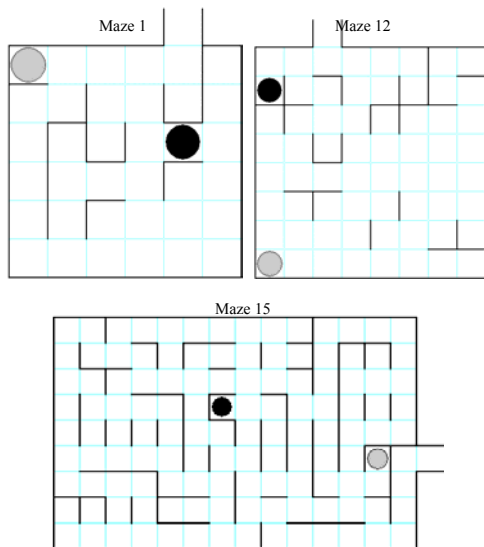


Figure 1. Sample *Theseus and the Minotaur* Mazes [[1]]

### 3 PROGRESSIVE REINFORCEMENT LEARNING

As stated earlier, basic RL approaches solve individual problems rather than classes of problems; they do not transfer knowledge gained in solving one problem to unseen but related problems. The approach that we introduce here, *Progressive Reinforcement Learning*, provides a mechanism for knowledge transfer. The essential characteristic of Progressive RL is that the agent is informed by alternating cycles of *experimentation* and *introspection*. In the very first experimentation phase, the agent has no prior experience and so solves a relatively simple problem using conventional RL methods. Having mastered the simple problem, it engages in a bout of introspection, in which it analyses the solution(s) found and generates a symbolic representation of the

solution, based on high-level features of the problem. In a subsequent experimentation phase with a more complex problem, this symbolic representation is used for those states about which the Reinforcement Learner has no information because they have not been explored. The intuition here is that experience based on simpler problems is only of limited use and may even be misleading, but is better than acting at random when venturing into areas of the current problem state-space that are unknown.

Figure 2 provides a schematic overview of how Progressive RL works. The phases of *experimentation* are conducted using the Reinforcement Learner, and the phases of *introspection* are conducted using the Symbolic Learner. This approach is not tied to a specific form of RL or a specific symbolic learning algorithm. The following sub-sections provide details of how these two major components of Progressive RL are implemented for the experiments reported below in Section 4.

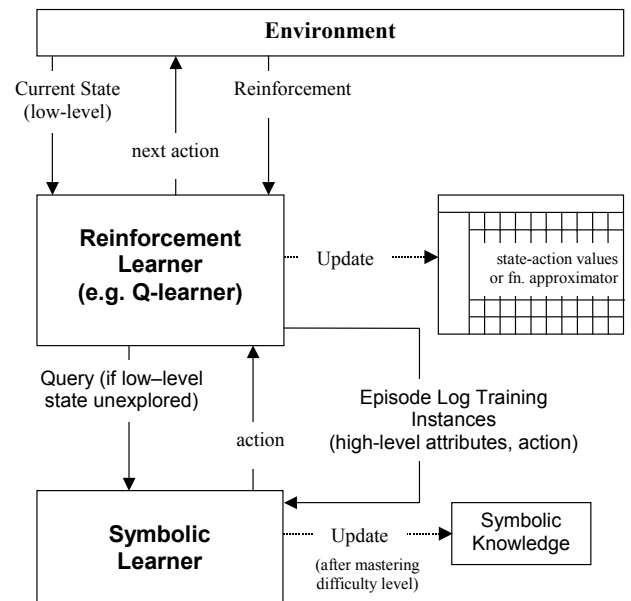


Figure 2. Architecture for Progressive Reinforcement Learning

#### 3.1 The Reinforcement Learner

The current implementation of Progressive RL is based on tabular Q-learning [25]. Table 1 provides details of the algorithm used in the experimentation phase of Progressive RL. The notation used in this description is the same as that used in Section 1. The main additional parameter is *BIAS*, described below. As shown on lines 2 and 14, a vector *Explored(s)* is maintained, indicating whether a state has been visited or not. If the current state is unexplored, a high-level description of it is passed to the Symbolic Learner, which in turn returns an action (lines 6-7). The Q-value for the chosen action is updated using the standard Q-learning approach and the Q-values for all other actions are set to the value *BIAS*. The next time this state is encountered, this *BIAS* value will encourage selection of the same rule-based action again. The need for this is in part due to the use of a default negative reward; in many cases, the chosen action will be updated to a negative value and, if

alternative actions had a value default of 0, that action would be avoided the next time that state is encountered. *BIAS* must not, however, be set to too large a negative number. Otherwise, when the Symbolic Learner chooses a bad action, the Q-values for other actions of that state will never catch up with the Q-value for the bad action and it will always be selected. In Section 4, the effect of *BIAS* is demonstrated experimentally.

**Table 1:** The Experimentation Phase Implemented in Progressive RL

---

```

1  Assume that Introspection Phase has built a set of rules from prior
   experience
2   $\forall s, \forall a: Q(s,a) = 0, \text{Explored}(s) = \text{false}$ 
3  Repeat (for each episode):
4    Initialise  $s$ 
5    Repeat (for each step of episode):
6      If Not Explored( $s$ ):
7         $a = \text{SymbolicLearner.ChooseAction}(\text{Highlevel}(s))$ 
8         $\forall a_i | a_i \neq a: Q(s, a_i) = \text{BIAS}$ 
9      Else
10      $a = \arg \max_a Q(s, a)$ 
11     With probability  $\epsilon: a = \text{random action}$ 
12     Take action  $a$ , observe reward  $r$  and next state  $s'$ 
13      $Q(s,a) = Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s,a)]$ 
14     Explored( $s$ ) = true
15      $s = s'$ 
16   Until  $s$  is terminal
17  Until stable solution found

```

---

**Table 2:** The Introspection Phase Implemented in Progressive RL

---

```

1  Assume that Experimentation Phase has reached steady state
2  Repeat (for a fixed number of episodes):
3    Follow Experimentation Phase procedure (lines 4-16)
4    When each action is taken:  $\text{Log}(\text{Highlevel}(s), a)$ 
5  End Repeat
6  Build Symbolic Learner mapping states to actions from logged
   instances

```

---

### 3.2 The Symbolic Learner

Table 2 outlines the algorithm used in the Introspection Phase. As described earlier, the role of the Symbolic Learner is to propose a course of action in situations that the Reinforcement Learner has not encountered previously, on the basis that experience gained on related (perhaps simpler) problems should be better than acting at random when venturing into areas of the current problem state-space that are unknown. However, the state-space and action policy used by the Reinforcement Learner in a previous problem would not directly correspond to that of the current problem, except where problems are only trivially different. Accordingly, the Symbolic Learner operates in terms of higher-level descriptions of the state space. This achieves two aims: it allows generalisation in associating actions with states in a specific problem, and it allows abstraction in applying knowledge gained in one problem to other related problems.

After an experimentation phase has reached steady state, several episodes are carried out according to the standard experimentation procedure, except that a high-level description of each state and the corresponding action chosen is recorded, as shown in Table 2. These records are then used by the Symbolic Learner to construct rules that map state descriptions to actions. The rules are in effect generated from the learned Q-values of the experimentation phase.

This approach is a form of behavioural cloning (see, for example, Šuc [15]) but it does not suffer the problems caused by learning from positive examples only, as the rules are generated from the actions taken when an optimal policy is being followed (steady-state).

Naturally, the two most important issues for the operation of the Symbolic Learner are the learning algorithm and the state descriptions. For the experiments reported in this paper, two learning algorithms are evaluated: Naive Bayes [11] and C4.5 [16]. In both cases, the implementations of these algorithms in the WEKA machine learning package [26] are used. The performance of the two classifiers in this domain is discussed in Section 4.

In future work, it is intended to explore techniques for automatic generation of high-level state descriptions. However, for the present experiments on the *Theseus* mazes, a fixed set of features has been chosen manually:

1. Distance to Minotaur: Manhattan distance
2. Distance to Exit: Manhattan distance
3. Wall to the West: [true,false]
4. Wall to the North: [true,false]
5. Wall to the East: [true,false]
6. Wall to the South: [true,false]
7. Wall to the West of Minotaur: [true,false]
8. Wall to the North of Minotaur: [true,false]
9. Wall to the East of Minotaur: [true,false]
10. Wall to the South of Minotaur: [true,false]
11. Direction to Minotaur: [N, NE, E, SE, S, SW, W, NW]
12. Direction to Exit: [N, NE, E, SE, S, SW, W, NW]

## 4 EXPERIMENTS AND RESULTS

The objective of these experiments is to compare the performance of Progressive RL with that of standard Q-learning, to see whether its knowledge transfer leads to improvements in later mazes. In these experiments, Progressive RL used cumulative rule building: following the experimentation phase on Maze  $n$ , the introspection phase considered experience gained in Mazes 1- $n$ . One experimentation phase corresponded to a fixed number of winning episodes of one maze.

Figure 3 and Figure 4 compare the performance on all mazes of basic Q-learning with two versions of Progressive RL: the first using C4.5 [16] as the Symbolic Learner and the second using Naive Bayes [11]. Figure 3 shows the number of episodes required to solve each maze (win) for the first time, averaged over 20 test runs, where an episode starts in a fixed initial state and ends in a win, lose or stalemate. The same parameter values were used in all experiments:  $\alpha = 0.3$ ,  $\gamma = 0.9$ ,  $\epsilon = 0.01$ ,  $r_n = -0.02$  and  $\text{BIAS} = -0.2$ .

The results for Maze 1 are identical for all approaches, since Progressive RL has not yet carried out the first phase of introspection. From Maze 2 on, however, both versions of Progressive RL require a significantly smaller number of episodes in getting the first win of each maze. Both versions of Progressive RL show similar levels of speed-up over standard Q-learning on average: 38.5% using C4.5 and 39.1% using Naive Bayes. Progressive RL also performs better than standard Q-learning over the course of entire experimentation phases, as shown in Figure 4, which graphs the average total reward received during each full experimentation phase (4000 wins) on each maze. Similar improvements were found in the number of episodes per win and number of steps taken per win over each experimentation phase.

Additional tests have shown that the experience gained from performing the experimentation phase on a difficult maze (e.g. Maze 15) can be transferred to simpler mazes and outperform standard Q-learning.

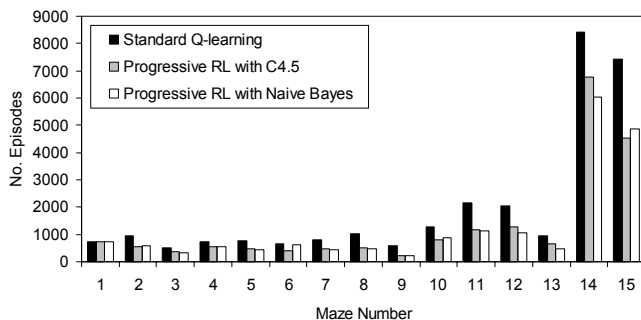


Figure 3. Comparison of Q-learning with Progressive RL: Number of Episodes to First Win

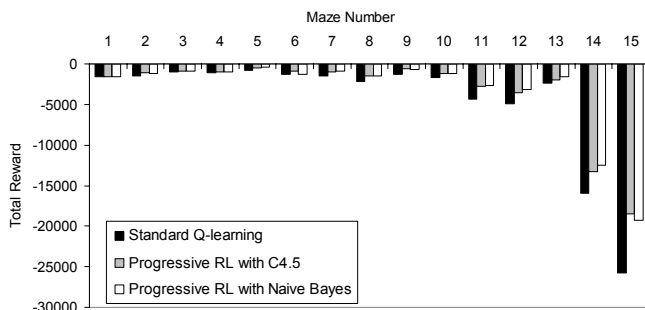


Figure 4. Comparison of Q-learning with Progressive RL: Total Reward per Experimentation Phase

In order to find a suitable value for BIAS, a parameter search was performed: Progressive RL was run on the set of mazes with different values for BIAS and the value  $-0.2$  was found to give the best results. Figure 5 illustrates the effect of BIAS, comparing the performance of standard Q-learning with Progressive RL on a single maze (Maze 15), averaged over 20 test runs. In these results, Naive Bayes is used as the Symbolic Learner, and one version has *BIAS* set to  $-0.2$  and the other version has *BIAS* set to  $0$ . It may be seen that even without *BIAS*, Progressive RL gains an initial advantage over standard Q-learning in completing the first win. This advantage is lost, however, as the learner without *BIAS* requires more steps to complete the next three wins before it finds an optimal solution. The reason for this is that after the first win in this case, the Q-values of many of the actions chosen by the Symbolic Learner are lower than the other actions of the same state. More learning must therefore be carried out in these states, which has the net effect of delaying the discovery of the optimal solution. In contrast, Progressive RL with *BIAS* set to  $-0.2$  finds the optimal solution after the first win of the maze.

Figure 6 compares the performance of these three learners after a stable solution has been found. This graph, which plots the number steps per win over 100 wins, shows that all the learners settle down to a similar behaviour after a stable solution has been found.

Overall, Progressive RL provides a significant speed-up in the initial discovery of a solution, relative to standard Q-learning.

Similar improvement levels were found when Sarsa [20] was used as the Reinforcement Learner (though its base performance was not as good as that of Q-learning).

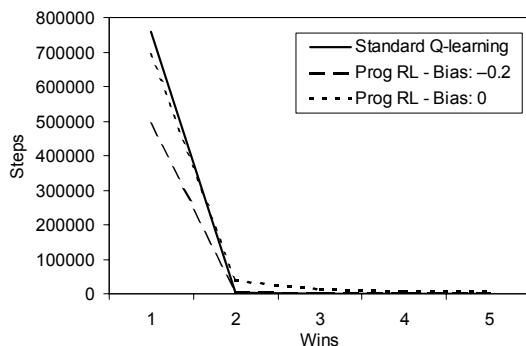


Figure 5. Effect of Bias on Progressive RL: First 5 Wins

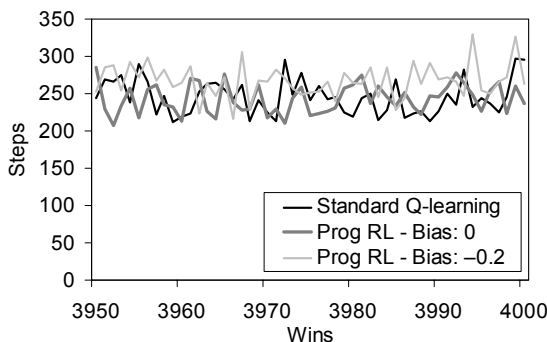


Figure 6. Effect of Bias on Progressive RL: Steady State

## 5 RELATED RESEARCH

### 5.1 Symbolic Knowledge in Reinforcement Learning

While Tesauro [22] demonstrated how to achieve state-space generalisation by using a neural network, other researchers have considered symbolic representations of Q-functions. The survey by Kaelbling *et al.* [10] discusses several approaches to generalisation. More recently, Džeroski, De Raedt and Driessens propose Relational Reinforcement Learning [9], in which the Q-function is represented as a logical regression tree [3]. The nodes in logical regression trees are first order logic tests, rather than simple propositional logic tests as used in classical decision trees (e.g. [16]). The main benefits of this approach are that it allows state-space abstraction and can represent structural aspects of the environment.

## 5.2 Cognitive Skill Acquisition

Breisemeister *et al.* [4] [5] propose a model for cognitive skill acquisition that comprises four basic components: a problem solver, a concept formation component (incorporating a decision tree learning algorithm), knowledge structure (decision tree itself) and a state transformation loop. This system is used in two phases: preparatory and application. In the preparatory phase, the problem solver, an A\* search algorithm, is applied to a given number of initial states (a training set). The problem solver generates a number of state-control action pairs that are passed to the learning component, which in turn induces a decision tree that classifies the set of states. During the application phase, the system tries to solve new problems by using the stored decision tree first. If the actual state cannot be classified or a cycle appears, the whole task is passed on to the problem solver. The problem solver then produces a solution, which, as in the preparatory phase, is also processed by the learner.

This work is of interest as it presents a model in which knowledge gained by the problem solver is used in future similar tasks by accessing the current decision tree structure. It is not clear, however, how the model generalises the state space in, for example, the problem of an agent going to a grocery store to buy items from a shopping list [5]. In their approach, each situation or state is a combination of the current grid square  $(x,y)$  and the list of goods remaining. The decision tree learner uses these  $x$ - $y$  coordinates as attributes, suggesting that knowledge gained is only applicable to identical store layouts.

## 5.3 Hybrid Reinforcement Learners

CLARION [17] [18] [19] is a two-level model that uses Q-learning at the lower level to gain procedural knowledge and uses of a set of symbolic propositional rules (declarative knowledge) at the upper level. Essentially, the approach is to operate standard RL and symbolic RL (discussed earlier in Section 5.1) in parallel, using a weighted sum to arbitrate between the actions proposed by the two techniques. The approach was applied to a maze problem [17] and to a simulated underwater mine navigation task [18]. It was concluded that CLARION outperformed basic Q-learning in the navigation task [18], and that rule induction facilitated transfer between problems where boundaries or obstacles were changed.

The CLARION architecture, like our Progressive RL architecture, has Q-learning and rule learning components. The principle difference is that CLARION consults both the rules and the Q-values in order to decide every action, combining their recommendations with a weighted sum, whereas Progressive RL uses symbolic knowledge only for unexplored states. Because of this difference, CLARION updates both the rules and Q-values at each step, whereas Progressive RL only constructs symbolic knowledge (in an introspection phase) when an experimentation phase is complete.

Dixon *et al.* [7] propose a more general hybrid model than that of CLARION, in which an *exploration control module* is used to couple a reinforcement learner with one or several sources of prior knowledge. The control module arbitrates between their recommendations, using an arbitration policy that may be varied. They demonstrate the approach in two domains where complex tasks are manually decomposed into simpler sub-tasks, and Q-learners trained on the sub-tasks are used as the prior knowledge for the overall tasks. Their results show that (as might be expected)

this decomposition results in significantly faster learning than standard Q-learning.

## 5.4 Other Techniques for Handling Complexity

One of the motivations behind the work described in this paper is to develop techniques that are able to tackle complex problems more efficiently, by first gaining experience at less complex problems and then transferring that experience to problems that are more complex. Other researchers have investigated alternative approaches to handling complexity in RL. The brief but useful introduction to RL by Russell and Norvig [14] identifies two areas of active research that aim to improve the effectiveness of RL algorithms at tackling complex problems:

1. Reward Shaping, in which rewards are provided for making progress (e.g. Ng *et al.* [13])
2. Hierarchical RL, in which large tasks are broken into subtasks which are tackled at a level where they are tractable; this includes research by Dietterich [6], Sutton *et al.* [21], and Andre and Russell [2].

The SKILLS algorithm of Thrun and Schwartz [[24]] is related to Hierarchical RL. In that algorithm, a skill is a partial action policy that is defined for a subset of all skills (the domain of the skill). Because the skills are defined for a region of the state-space rather than the whole state-space, they are applicable to entire sets of related tasks, rather than individual tasks. However, it is assumed that all related tasks have identical states and actions. Accordingly, the SKILLS approach would not be directly applicable to the *Theseus* environment.

Clearly, these approaches are quite different from the approach of Progressive RL. This means, of course, that it might be possible to gain additional improvements in handling complexity by adopting a mixed strategy, for example combining Hierarchical RL with Progressive RL.

## 6 CONCLUSIONS & FUTURE WORK

This paper has introduced Progressive Reinforcement Learning, which augments standard Q-Learning with a mechanism for transferring experience gained in one problem to a new, related, problem. Accordingly, this work falls into the area of lifelong learning, as originally identified by Thrun [23].

Progressive RL may be considered to implement a simplified version of the cognitive model of progression from novice to expert, proposed by Dreyfus and Dreyfus [8]. In our approach, an agent acquires experience of operating in a simple domain through *experimentation*. It then engages in a period of *introspection*, during which it rationalises the experience gained and formulates rules describing how to behave in that simple domain. When subsequently experimenting in a more complex but related domain, it is guided by the rules until it gains direct experience. Our experiments demonstrate the benefit of Progressive RL in a sequence of 15 maze puzzles, relative to a basic RL approach in which each puzzle is solved from scratch.

This research is in its early stages. In the current implementation, the RL and Symbolic Learner components are quite basic, and there are many other options to be considered. Also, limitation of the current approach is that the high-level features used by the symbolic learner are manually selected, so we

hope to develop schemes that are more general for producing such features. Another aspect of this approach that may merit future research is that it would support manually specified rules (e.g. "you cannot walk through walls") as well as the automatically-discovered ones. Finally, we intend to develop a suite of different RL test environments, each of which will have progressive levels of difficulty.

## ACKNOWLEDGEMENTS

This research has been supported by NUI Galway's Millennium Research Fund. The authors are grateful to Mr. Robert Abbott and Mr. Toby Nelson for their permission to use the *Theseus and the Minotaur* mazes.

## REFERENCES

- [1] Abbott, R: Mad Mazes: Intriguing Mind Twisters for Puzzle Buffs, Game Nuts and Other Smart People. Adams Media, 1990 (<http://www.logicmazes.com>).
- [2] Andre, D and Russell, SJ: State Abstraction for Programmable Reinforcement Learning Agents. Proc. 18<sup>th</sup> National Conference on Artificial Intelligence, 2002.
- [3] Blockeel, H and De Raedt, L: Top-Down Induction of First Order Logical Decision Trees. Artificial Intelligence, Vol. 101(1-2), pp. 285-297, 1998.
- [4] Breisemeister, L, Scheffer, T and Wysotzki, F: Combination of Problem Solving and Learning from Experience, Technical Report 20/95, TU Berlin, 1995.
- [5] Breisemeister, L, Scheffer, T and Wysotzki, F: A Concept Formation Based Algorithmic Model for Skill Acquisition, Proc. First European Workshop on Cognitive Modelling, 1996.
- [6] Dietterich, T: Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. Journal of Artificial Intelligence Research, Vol 13, 2000.
- [7] Dixon, KR, Malak, RJ and Khosla, PK: Incorporating Prior Knowledge and Previously Learned Information into Reinforcement Learning Agents. Technical Report, Institute for Complex Engineered Systems, CMU, 2000.
- [8] Dreyfus, HL and Dreyfus, SE: Mind Over Machine: The Power of Human Intuition and Experience in the Era of the Computer. Blackwell, 1986.
- [9] Džeroski, S, De Raedt, L and Driessens, K: Relational Reinforcement Learning. Machine Learning, Vol. 43, pp. 7-52, 2001.
- [10] Kaelbling, LP, Littman, ML and Moore, AW: Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research, Vol. 4, pp. 237-285, 1996.
- [11] Langley, P, Iba, W and Thompson, K: An Analysis of Bayesian Classifiers. Proc. 10<sup>th</sup> National Conference on Artificial Intelligence, pp 223-228, 1992.
- [12] Michie, D, Bain, M and Hayes-Michie, J: Cognitive Models from Subcognitive Skills. Knowledge-Based Systems in Industrial Control, pp 71-99, 1990.
- [13] Ng, AY, Harada, D and Russell, SJ: Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. Proc. 16<sup>th</sup> International Conference on Machine Learning, 1999.
- [14] Russell, SJ and Norvig, P: Artificial Intelligence: A Modern Approach. Prentice Hall, 2003 (Second Edition).
- [15] Šuc, D.: Skill Machine Reconstruction of Human Control Strategies, Ph.D. Thesis, University of Ljubljana, Slovenia, 2001.
- [16] Quinlan, JR: C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
- [17] Sun, R, Peterson, T and Merrill, E: Bottom-up Skill Learning in Reactive Sequential Decision Tasks. Proc. 18<sup>th</sup> Cognitive Science Society Conference, 1996.
- [18] Sun, R and Peterson, T: Autonomous Learning of Sequential Tasks: Experiments and Analyses. IEEE Transactions on Neural Networks, Vol. 9, pp. 1217-1234, 1998.
- [19] Sun, R and Merrill, E: From Implicit Skills to Explicit Knowledge: A Bottom-Up Model of Skill Learning. Cognitive Science, Vol. 25, No. 2, 2001.
- [20] Sutton, RS and Barto, AS: Reinforcement Learning, an Introduction. MIT Press, 1998.
- [21] Sutton, RS, McAllester, DA, Singh, SP and Mansour, Y: Policy Gradient Methods for Reinforcement Learning with Function Approximation. Proc. Advances in Neural Information Systems, 2000.
- [22] Tesauro, G: Practical Issues in Temporal Difference Learning. Machine Learning, Vol. 8, pp. 257-277, 1992.
- [23] Thrun, S: Explanation-Based Neural Network Learning: A Lifelong Learning Approach. Kluwer Academic Publishers, 1996.
- [24] Thrun, S and Schwartz, A: Finding Structure in Reinforcement Learning. Proc. Advances in Neural Information Systems, 385-392, 1995.
- [25] Watkins, CJCH: Learning from Delayed Rewards. Ph.D. Thesis, Cambridge University, 1989.
- [26] Witten, IH and Frank, E: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers, 2000.