Artificial Intelligence Review **21:** 375–398, 2004.

Transfer of Experience between Reinforcement Learning Environments with Progressive Difficulty

MICHAEL G. MADDEN & TOM HOWLEY
Department of Information Technology, National University of Ireland, Galway, Ireland (E-mail: michael.madden@nuigalway.ie)

Abstract. This paper describes an extension to reinforcement learning (RL), in which a standard RL algorithm is augmented with a mechanism for transferring experience gained in one problem to new but related problems. In this approach, named Progressive RL, an agent acquires experience of operating in a simple environment through *experimentation*, and 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 environment. When subsequently experimenting in a more complex but related environment, it is guided by this knowledge until it gains direct experience. A test domain with 15 maze environments, arranged in order of difficulty, is described. A range of experiments in this domain are presented, that demonstrate the benefit of Progressive RL relative to a basic RL approach in which each puzzle is solved from scratch. The experiments also analyse the knowledge formed during introspection, illustrate how domain knowledge may be incorporated, and show that Progressive Reinforcement Learning may be used to solve complex puzzles more quickly.

Keywords: C4.5, experience transfer, Naive Bayes, PART, Progressive RL, Q-learning, reinforcement learning, rule learning

1. Introduction

Reinforcement learning (RL) may be considered to be 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. (Utgoff and Cohen, 1998, offer some observations about when RL is not suitable.) Indeed, Dreyfus and Dreyfus (1986) 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 for many 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.

It is assumed that the reader is familiar with the basics of reinforcement learning. Russell and Norvig (2003) give a useful summary, and a more comprehensive introduction is provided by Sutton and Barto (1998).

This paper is structured as follows. Section 2 provides a context for this work by presenting a set of maze problems called the *Theseus and the Minotaur* mazes (Abbott, 1990). 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 RL approach, which seeks to augment standard Q-learning (Watkins, 1989; Sutton and Barto, 1998) with a mechanism for transferring experience gained in one problem to a new, related, problem. A comprehensive set of experiments in applying Progressive RL to the *Theseus and the Minotaur* mazes are presented and discussed in Section 4. Then, Section 5 describes related research. Finally, conclusions are drawn 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 (Abbott, 1990). Figure 1 shows a selection of the mazes. In them, 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, where complexity is determined by the size of the decision space as well as the number of steps to the solution.

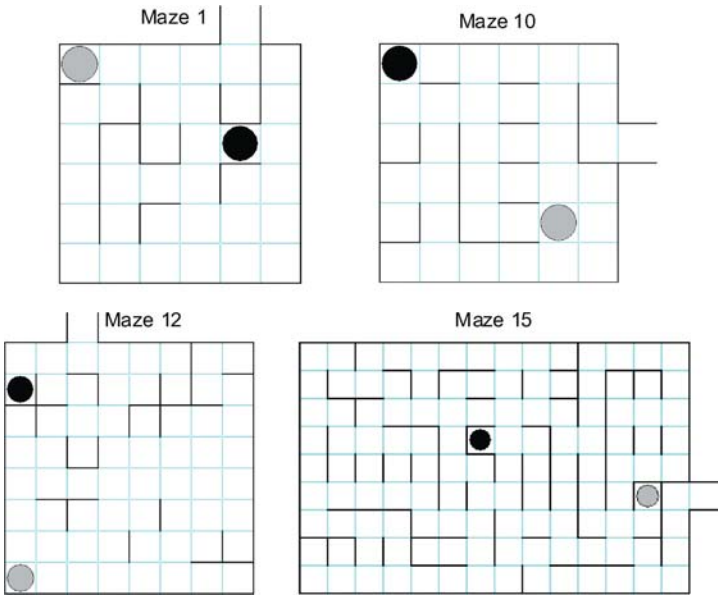

Figure 1. Sample *Theseus and the Minotaur* Mazes (Abbott, 1990).

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 (Watkins, 1989) 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 five actions: North, East, South, West and Delay. A lookup table of Q-values is maintained, e.g. for Maze 1 (6×6) a table of 6480 entries (5 actions $\times$ 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, α the step-size parameter, γ the discount factor and $\max_{a'} Q(s', a')$ 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, if the same state and action path is traversed on two consecutive moves, then the play is ended and a reward of -1 is given. Exploration is achieved by following an ε-greedy policy, in which the action with the highest Q-value is chosen with probability $1-\varepsilon$.

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

3. Progressive Reinforcement Learning

As stated in the Introduction, basic RL approaches solve individual problems rather than classes of problems; they do not transfer knowledge gained in solving one problem to other related problems. The approach that we introduce here, *Progressive Reinforcement Learning*, provides a mechanism for knowledge transfer. Its essential characteristic 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 a reinforcement learner, and the phases of *introspection* are conducted using a symbolic learner. This approach is not tied to a specific form of RL or a specific symbolic learning algorithm. The following sub-sections provide

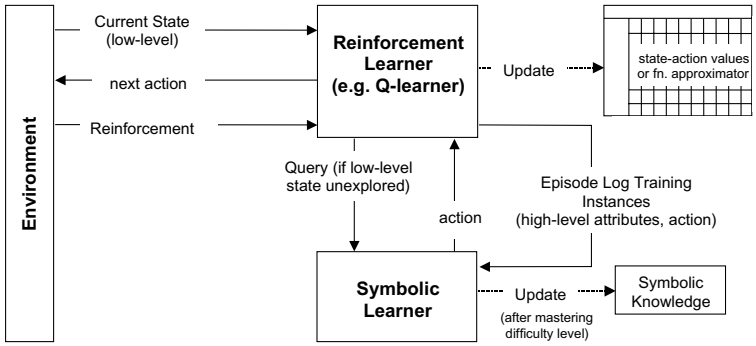

Figure 2. Architecture for Progressive Reinforcement Learning.

details of how these two major components of Progressive RL are implemented for the experiments reported below in Section 4.

3.1. *The reinforcement learner*

In the current implementation of Progressive RL, the experimentation phase uses tabular Q-learning (Watkins, 1989). 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 2. 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 has been explored, then a standard RL strategy is used (lines 9–11). However, if it is unexplored, a high-level description of it is passed to the symbolic learner, which in turn returns an action (lines 6–8). 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 4.2, 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	$\forall s, \forall a$: $Q(s,a) = 0$, Explored(s) = false
3	Repeat (for each episode):
4	Initialise s
5	Repeat (for each step of episode):
6	If Not Explored(s):
7	a = SymbolicLearner.ChooseAction(Highlevel(s))
8	$\forall a_i \mid a_i \neq a$: $Q(s, a_i) = BIAS$
9	Else
10	$a = \arg\max_a Q(s, a)$
11	With probability ε a = *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: Log(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, during which 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.

The approach followed in the introspection phase is related to behavioural cloning (see, for example, Šuc, 2001). However, 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, when the prior experimentation phase has reached a steady-state solution.

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, three different learning algorithms are evaluated, as discussed later in Section 4. In all three cases, the implementations of these algorithms in the WEKA machine learning package (Witten and Frank, 2000) are used.

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 quite basic features has been chosen manually. These are listed in Table 3.

4. Experimental Analysis

This section provides a comprehensive experimental analysis of Progressive RL. In the first sub-section, its performance is compared with that of a benchmark RL algorithm. In the two sub-sections immediately after, specific implementation details of Progressive RL are examined: the use of the BIAS parameter and the use of cumulative experience during introspection. Then, in Section 4.4, the knowledge discovered during an introspection phase is examined. After that, Section 4.5 demonstrates how Progressive RL facilitates the incorporation of

Table 3. High-level features used to describe environments in the experiments

Name	Description	Units
1 `WallWest`	Wall to the West	[true, false]
2 `WallNorth`	Wall to the North	[true, false]
3 `WallEast`	Wall to the East	[true, false]
4 `WallSouth`	Wall to the South	[true, false]
5 `WallWestMinotaur`	Wall to the West of Minotaur	[true, false]
6 `WallNorthMinotaur`	Wall to the North of Minotaur	[true, false]
7 `WallEastMinotaur`	Wall to the East of Minotaur	[true, false]
8 `WallSouthMinotaur`	Wall to the South of Minotaur	[true, false]
9 `DistMinotaur`	Distance to Minotaur	Manhattan distance
10 `DistExit`	Distance to Exit	Manhattan distance
11 `DirMinotaur`	Direction to Minotaur	[N, NE, E, SE, S, SW, W, NW]
12 `DirExit`	Direction to Exit	[N, NE, E, SE, S, SW, W, NW]

domain knowledge (or instruction). Finally, Section 4.6 illustrates how Progressive RL may be viewed as a technique for reducing complexity when solving large problems.

4.1. *Comparing Progressive RL with a standard RL algorithm*

The objective of this first group of 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. It is felt that this is an appropriate benchmark algorithm because the current implementation of Progressive RL is based on standard Q-learning. However, the Progressive RL architecture is in theory independent of any specific RL algorithm. Thus, while a more sophisticated RL algorithm might perform better than Q-learning for this *Theseus* domain, such an algorithm could equally be incorporated into Progressive RL. Accordingly, there is no clear case for benchmarking Progressive RL against such an algorithm.

In this group of experiments, Progressive RL uses cumulative rule building, meaning that after the experimentation phase on Maze n, the introspection phase considers experience gained in all Mazes 1 to n. One experimentation phase corresponds to a fixed number of winning episodes of one maze. Figures 3 and 4 compare the performance on all

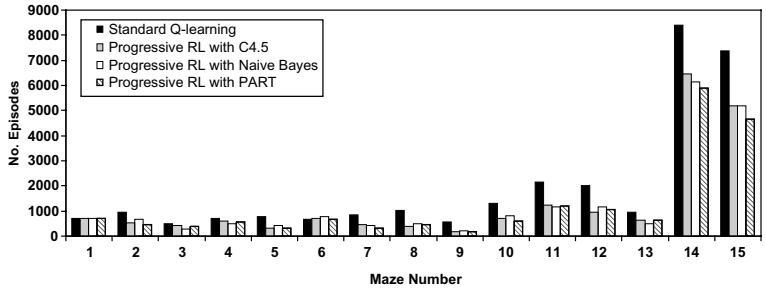

Figure 3. Comparison of Q-learning with Progressive RL: average number of episodes to first win.

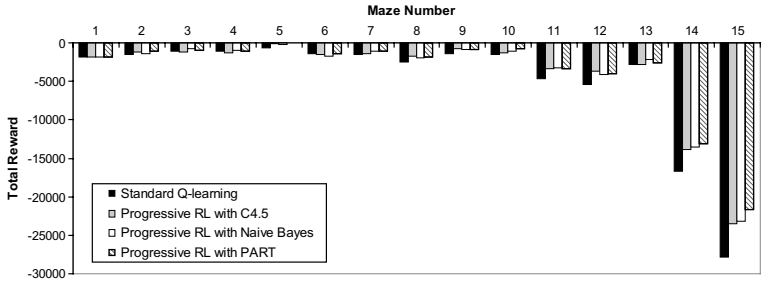

Figure 4. Comparison of Q-learning with Progressive RL: average reward accumulated over 1000 wins.

mazes of basic Q-learning with three versions of Progressive RL, each using a different type of learner for the introspection phase:

1. C4.5 Decision Trees (Quinlan, 1993)
2. Naive Bayes (Langley et al., 1992)
3. PART Partial Decision Trees (as described by Witten and Frank, 2000)

Figure 3 shows the number of episodes required to solve each maze (win) for the first time, averaged over a set of 10 experiments, where an episode starts in a fixed initial state and ends in a win, lose or stalemate, and a single experiment involves working through all 15 mazes, solving each one 1000 times. The same parameter values were used in all experiments: $\alpha = 0.3$, $\gamma = 0.9$, $\varepsilon = 0.01$, $r_n = -0.02$ and 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, all three versions of Progressive RL require a significantly smaller number of episodes in getting the first win of each

maze, apart from Maze 6 for which only Progressive RL with PART shows a slight improvement over standard Q-learning. (It may be that Maze 6 requires a strategy not seen in the earlier mazes.) All versions of Progressive RL show significant levels of speed-up over standard Q-learning on average: 37.99% using C4.5, 38.24% using Naive Bayes and 42.07% using PART.

Progressive RL also performs better than standard Q-learning over the course of an entire set of experiments, as shown in Figure 4, which graphs the total reward accumulated over each full experiment (1000 wins) on each maze, averaged over the 10 experiments. (Note that these values are negative since the allocation of rewards in these environments, as described earlier in Section 2, generally results in negative rewards being accumulated by the learner.) Similar improvements were found in the number of episodes per win and number of steps taken per win over each experimentation phase (not graphed). 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, though the merits of such an approach are not very obvious.

Overall, Progressive RL provides a significant speed-up in the initial discovery of a solution, relative to standard Q-learning. Some experiments were also carried out using Sarsa (Sutton and Barto, 1998) as the reinforcement learner in the experimentation phase, and similar percentage improvements were found. However, the base performance of Sarsa was not as good as that of Q-learning for these environments, so the work presented in this paper focuses on Q-learning.

4.2. *Examining the effect of BIAS*

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

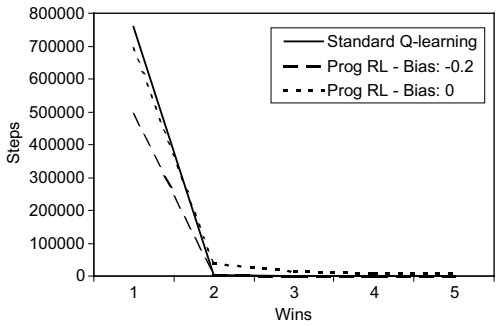

Figure 5. Effect of BIAS on Progressive RL: First 5 wins.

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 the course of 50 wins, shows that all the learners settle down to a similar behaviour after a stable solution has been found.

4.3. *Why base introspection on cumulative experience?*

The experiments described above in Section 4.1 use the cumulative experience gained in Mazes 1 to n in the introspection phase following experimentation on Maze n. It is reasonable to ask whether this is necessary, and whether anything substantially new is learned after Maze 1.

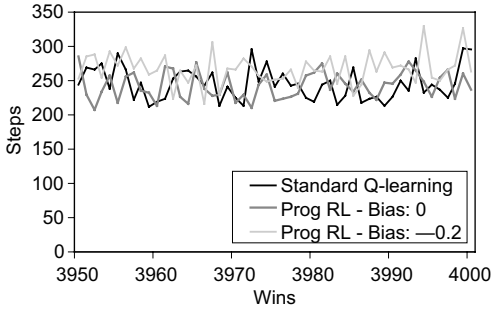

Figure 6. Effect of BIAS on Progressive RL: steady state.

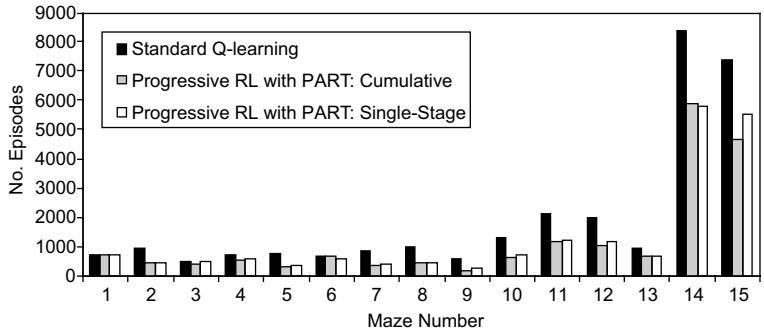

Figure 7. Performance of Progressive RL with rule building stopped after Maze 1: average number of episodes to first win.

In order to ascertain how much new knowledge is acquired after the introspection phase for the first maze has been completed, experiments were performed where the rule-building process was stopped after completion of the first maze. Cumulative Progressive RL (with PART as the symbolic learner) was compared with this single-stage version of Progressive RL. The results, shown in Figure 7, indicate that the cumulative approach shows some improvement in speed-up over the approach based only on the single-stage rules generated from executing Maze 1. Both versions of Progressive RL perform identically on Mazes 1 and 2, naturally. Over the remaining mazes (3–15) single-stage Progressive RL achieves an average speed-up of 35.07%, compared with 41.3% for the cumulative version,[1] in both cases relative to standard Q-learning.

This result demonstrates that, for the *Theseus* set of environments, experience gained in a simple environment such as Maze 1 continues to be useful in much more complex environments such as Maze 15. Nonetheless, the result indicates that there is still some benefit in terms of speed-up in using the cumulative approach. Also, as reported below in Section 4.4, the knowledge formed from cumulative experience of multiple environments will tend to reflect general principles rather than artefacts of one particular environment. Depending on the application and the algorithmic complexity of the underlying symbolic learner, it may be more appropriate to use either the cumulative or single-stage form of Progressive RL.

4.4. *Examining the knowledge formed during introspection*

The symbolic representation generated during the introspection phase of Progressive RL is of interest as it gives an insight into what is actually

being learned. In this sub-section, rules formed by the variant of Progressive RL that uses PART are examined, for two reasons: firstly, the results of Section 4.1 have shown that it is best over Mazes 2 to 15 in terms of average speed-up gained over standard Q-learning; secondly, the decision lists that PART generates are relatively comprehensible. Figure 8 shows a sample set of rules that were generated while running Progressive RL with PART on Maze 1. (In this and the next figure, the numbers in parentheses are the counts of instances that provide support for/against a rule; thus they indicate the importance and confidence level of a rule.) Note that these rules should be interpreted as an ordered list rather than being considered independently of each other. The first rule tells Theseus to move east if there is a wall blocking the south and west directions and if the distance to the Minotaur is less than 7. This rule helps Theseus to avoid walking into walls. However, some of these rules include checks of the distance to the Minotaur (`DistMinotaur`) that are not intuitively obvious, such as the fourth rule which returns 'north' provided the distance to the Minotaur is greater than 1 and less than or equal to 3. Such rules correspond to manoeuvres that are of use in specific parts of solving Maze 1, but are not more generally applicable.

Figure 9 presents PART rules based on the cumulative experience of successfully solving Mazes 1 to 14. In addition to ensuring Theseus does not walk into walls, the first two rules guide Theseus in the direction of the maze exit. In the first rule, for example, in which Theseus has the

```
WallSouth = true AND
WallWest = true AND
DistMinotaur <= 7: GO EAST (453.0/2.0)

WallSouth = true AND
WallNorth = true AND
DistMinotaur > 1: GO EAST (221.0/1.0)

WallSouth = true AND
WallWest = false AND
DistMinotaur > 4 AND
WallWestMinotaur = true: GO WEST (462.0/4.0)

WallSouth = true AND
DistMinotaur > 1 AND
DistMinotaur <= 3 AND
WallWestMinotaur = true: GO NORTH (334.0/5.0)
.
.
. (continuation of PART Decision List)
```

Figure 8. Sample of PART decision list after learning on Maze 1 (6568 instances).

```
WallEast = true AND
WallWest = true AND
WallSouthMinotaur = false AND
WallNorth = false AND
DirExit = north-east AND
WallSouth = false AND
WallEastMinotaur = true AND
DirMinotaur = west: GO NORTH (1022.0/7.0)

WallEast = true AND
WallWest = true AND
WallSouthMinotaur = false AND
WallNorth = false AND
DirExit = north: GO NORTH (719.0/7.0)

WallEast = true AND
WallWest = true AND
WallSouthMinotaur = true AND
DistMinotaur <= 3: GO SOUTH (2107.0/23.0)

WallEast = true AND
WallWest = true AND
WallNorth = true AND
DistMinotaur <= 7: GO SOUTH (341.0/1.0)
.
.
. (continuation of PART Decision List)
```

Figure 9. Sample of PART decision list after learning on Mazes 1 to 14 (94,271 instances).

option of moving north or south (both `WallSouth` and `WallNorth` are false) north is chosen, since the maze exit (`DirExit`) is to the north-east. The fourth rule, which covers the scenario where Theseus is blocked by a wall on all sides except the south-facing one, advises Theseus to move south as expected.

It may be noted that these cumulatively-learned rules are somewhat more general, and hence intuitively understandable, than the single-stage rules based on Maze 1 alone. As was mentioned in Section 4.3, a benefit of rules discovered in cumulative Progressive RL is that they are less likely to incorporate artefacts of a single environment and more likely to reflect general principles of the family of environments.

While it is interesting to be able to make sense of the rules discovered during introspection on the maze environments, it is clear that these rules are quite simple: they principally direct Theseus to avoid walls and move towards the exit, though some appear also to propose moves based on the distance to the Minotaur and its relative direction. The reason the rules are so simple is twofold: firstly, the feature set used for high-level state descriptions, as listed earlier in Table 3, are very basic;

and secondly, the symbolic learners that are used for introspection are capable only of representing propositional logic terms. Accordingly, the current implementation of Progressive RL is restricted in discovering complex strategies that might, for example, lead the Minotaur into a trap. In the future, it is hoped to explore methods for automatic generation of high-level state descriptions. It would also be possible to use a first-order logic learner such as FOIL (Quinlan, 1990) as the symbolic learner for introspection phases.

4.5. *Incorporating domain knowledge*

The Progressive RL architecture is designed to extract knowledge from past experience, thereby engaging in unsupervised learning or self-teaching. However, it may also make use of domain knowledge that is manually constructed by a human expert. This may be considered to be a simple model of instruction, as an alternative to self-teaching. Section 5.3 outlines some work by other researchers on this topic.

A set of experiments was carried out to demonstrate how to provide Progressive RL with set of manually constructed rules. In this set of experiments, an introspection phase is not performed. Instead, a fixed set of rules is built into the symbolic learner, reflecting elementary knowledge of the domain, and these rules are used to choose an action when an unexplored state is encountered. The following rules were used:

(1) If there is a Wall to the North of Theseus, do not go North
(2) If there is a Wall to the East of Theseus, do not go East
(3) If there is a Wall to the South of Theseus, do not go South
(4) If there is a Wall to the West of Theseus, do not go West
(5) If more than one action is left to be chosen, choose one randomly
(6) The 'Delay' action is never selected

These rules are expressed using the same high-level features (Table 3) that were used in earlier experiments, and can be considered to be representative of the rules that we would hope that standard Progressive RL could discover through introspection.

Progressive RL with these manually specified rules performs well, as can be seen in Figure 10, which shows the average number of episodes required to solve each maze. For comparison, the corresponding results for standard Q-learning and Progressive RL with PART are also shown. An overall speedup of 37.04% was achieved using manual rules in the symbolic learner. This is very close to the speedup of 37.99% achieved

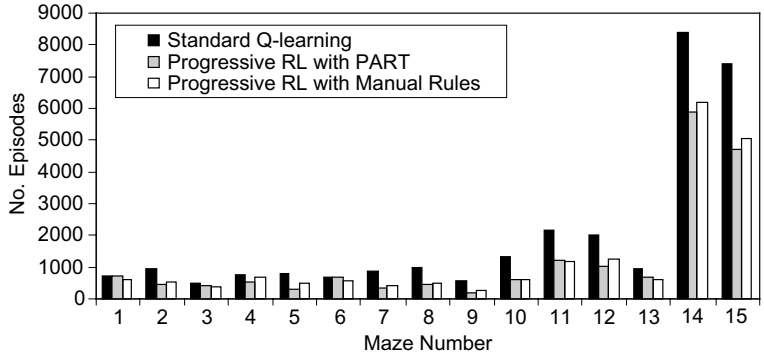

Figure 10. Comparison of Q-learning with Progressive RL using manual rules: average number of episodes to first win.

using Progressive RL with C4.5, as was reported in 4.1, though not quite as good as the speedup of 42.07% using PART. The extra improvement observed in Progressive RL with PART may be due to its consideration of the location of the maze exit or the location of the Minotaur.

Some observations may be made about this result. Firstly, it indicates the effectiveness of using domain knowledge (Progressive RL with manual rules) where it is known. On the other hand, it also indicates that, if domain knowledge is not available, Progressive RL with PART is able to discover useful knowledge through experimentation in simple environments. (An assumption underpinning all work in inductive learning is that it is generally easier to identify potentially important features than to formulate comprehensive knowledge in a domain. However, maze problems are readily understood by humans, so it is not particularly difficult to formulate useful rules in this particular domain.)

Finally, this result reflects the conclusion drawn at the end of Section 4.4, that the propositional logic-based learners and simple set of features used in this work do not enable the discovery of sophisticated problem-solving strategies.

4.6. *Progressive RL as a technique to reduce 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.

Results in the previous sub-sections, such as those of Figures 3 and 4, show that Progressive RL is preferable to a standard RL algorithm when

tackling a sequence of problems. In this sub-section, a slightly different question is addressed: *when faced with a complex problem, is it better to tackle it directly with a standard RL algorithm, or to first solve a related simple problem and apply the experience gained to the complex problem?*

To answer this question, it is first re-formulated as asking whether the benefit of using Progressive RL on a complex problem outweighs the overhead of first solving a simple problem. An experiment was performed wherein Progressive RL was used to solve the simplest maze (Maze 1) 20 times (to reach a stable solution) in a single experimentation phase, followed by an introspection phase that was based on the last ten solutions. In the next experimentation phase, each of the two most difficult mazes (14 and 15) were solved, independently of each other but using the knowledge gained from Maze 1. This was repeated 10 times and the results averaged. The experiment was repeated using each of the three symbolic learners that were listed in Section 4.1. For the purposes of comparison, the experiment was also repeated using standard Q-learning to solve to Mazes 14 and 15, though in this case solving Maze 1 was of course skipped.

Table 4 presents the results of this set of experiments. It lists the total number of episodes required to solve Mazes 14 and 15 for the first time, and the total reward accumulated over 10 runs of the experiment. The figures for the three versions of Progressive RL include the overhead from solving Maze 1. Thus, in counting the total number of episodes required to solve Maze 14 for the first time using Progressive RL, the

Table 4. Performance of Progressive RL on complex mazes, including overhead of first solving Maze 1, relative to Q-learning

	Maze 14	Maze 15
Total number of episodes to first win		
Q-Learning	8383.5	7386.4
Progressive RL (C4.5)	7579.5 (9.6%)	5443.0 (26.3%)
Progressive RL (Naive)	6491.6 (22.6%)	6112.9 (17.2%)
Progressive RL (PART)	6957.5 (17.0%)	5625.4 (23.8%)
Total reward per experiment		
Q-learning	−14,879.0	−22,911.9
Progressive RL (C4.5)	−13,439.6 (9.7%)	−15,927.0 (30.5%)
Progressive RL (Naive)	−11,993.3 (19.4%)	−18,031.3 (21.3%)
Progressive RL (PART)	−12,730.7 (14.4%)	−16,434.3 (28.3%)

number of episodes first required to solve Maze 1 twenty times are included, since solving Maze 1 was the first step in solving Maze 14. Likewise, the figures reported for total reward for Progressive RL include the (negative) reward accumulated from solving Maze 1 20 times. In the table, the numbers in parentheses are the improvements relative to the Q-learning results. In all cases, the improvements are substantial.

These results demonstrate that, for the *Theseus* puzzles, it is indeed beneficial to use Progressive RL to solve a simple problem first before tackling a more complex problem, rather than attempting to solve the more complex problem from scratch. This corresponds to the typical experience of humans tackling these puzzles, as mentioned in the Introduction, who find it easier to solve the large mazes after they have successfully solved the small ones.

5. Related research

5.1. *Knowledge transfer and symbolic knowledge in RL*

Several researchers have examined the transfer of knowledge or experience between related RL tasks. However, most of this work has focused on sets of problems where the state-space is essentially fixed, and the objective is to solve variations of a single problem. For example, Perkins and Precup (1999) and Bernstein (1999) deal with fixed distributions of Markov decision problems, and Carroll et al. (2001) considers problems in which the location of the goal is changed in each problem. Another approach that uses previously learned Q-values is the sub-problem reuse approach of Bowling and Veloso (1998; 1999), who present a method for determining the sub-optimality of reusing part of a previously learned policy. In this paper, the objective is to transfer experience from simple problems to more complex problems that have some similarities but do not have the same state-spaces. This means that learned policies such as Q-values cannot be directly reused.

Of more relevance to our work is the use of symbolic representations in RL. While Tesauro (1992) 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. (1996) discusses several approaches to generalisation. More recently, Džeroski et al. (2001) propose relational reinforcement learning, in which the Q-function is represented as a logical regression tree (Blockeel and DeRaedt, 1998). The nodes in logical regression trees are

first order logic tests, rather than simple propositional logic tests as used in classical decision trees. 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. (1995; 1996) 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 (the 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 if 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 (Breisemeister et al., 1996). In their approach, each situation or state is a combination of the current grid coordinate (x, y) and the list of goods remaining. The decision tree learner uses these coordinates as attributes, suggesting that knowledge gained is only applicable to identical store layouts.

5.3. *Hybrid reinforcement learners and domain knowledge*

CLARION (Sun et al., 1996; Sun and Peterson, 1998; Sun and Merrill, 2001) is a two-level model that uses Q-learning at the lower level to gain procedural knowledge and uses a set of symbolic propositional rules (declarative knowledge) at the upper level. Essentially, the approach is to operate standard RL and symbolic RL (as discussed in Section 5.1) in parallel, using a weighted sum to arbitrate between the actions proposed by the two techniques. The CLARION architecture, like our Progres-

sive RL architecture, has Q-learning and rule learning components. The principal 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.

CLARION was applied to a maze problem (Sun et al., 1996) and to a simulated underwater mine navigation task (Sun and Peterson, 1998). It was concluded that it outperformed basic Q-learning in the navigation task (Sun and Peterson, 1998), and that rule induction facilitated transfer between related problems where boundaries or obstacles were changed. However, its architecture as presented does not appear to be applicable to knowledge transfer where the state-space is changed.

Dixon et al. (2000) 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.

Dixon et al. (2000) also review related research in incorporating various forms of domain knowledge into reinforcement learning architectures, such as the system of Maclin and Shavlik (1996) that allows the user to provide advice in the form of commands, and the learning with an external critic (LEC) approach of Whitehead (1991). All of those approaches bear some similarity to the technique presented in Section 4.5 for incorporating domain knowledge into RL. Unlike Progressive RL, however, those approaches do not combine externally supplied knowledge with knowledge discovered through unsupervised experimentation.

5.4. *Other techniques for handling complexity*

As discussed in Section 4.6, Progressive RL may be used 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 (2003) 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., 1999)
2. Hierarchical RL, in which large tasks are broken into subtasks that are tackled at a level where they are tractable; this includes research by Dietterich (2000), Sutton et al. (2000), and Andre and Russell (2002).

The SKILLS algorithm of Thrun and Schwartz (1995) 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

This paper has described progressive reinforcement learning, which augments any standard reinforcement learning algorithm such as 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 (1996).

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 (1986). 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.

A range of experiments has been presented in this paper. The first set of experiments demonstrate the performance benefit of Progressive RL

in solving a sequence of 15 maze puzzles, relative to a basic RL approach in which each puzzle is solved from scratch. Some implementation details of Progressive RL have also been examined, specifically the use of the BIAS parameter and the use of cumulative experience during introspection. The form of knowledge that is discovered during an introspection phase has also been analysed, and other experiments have shown how the Progressive RL architecture facilitates the incorporation of domain knowledge. Finally, results have been presented that illustrate how Progressive RL may be viewed as a technique for reducing complexity when solving large problems, by using experience that has been gained in solving less complex problems.

Some potential avenues of future research have been identified in this paper. In particular, in Section 4.4 it has been observed that the introspection phase cannot devise complex solution strategies because it is limited by the simple nature of the high-level features that have been used, and by its use of symbolic learners based on propositional logic. This latter limitation could be overcome by using a first-order logic learner such as FOIL (Quinlan, 1990), but it is believed that this would not result in better results without more elaborate high-level features. Accordingly, it is hoped to explore methods for automatic generation of high-level state descriptions in the future, though this is by no means a trivial task. Finally, it is also intended that a suite of different RL test environments will be developed, 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 for his permission to use the *Theseus and the Minotaur* mazes.

Notes

[1] This differs to the value of 42.07% reported in Section 4.1 because it does not include the results of Maze 2 in this case.

References

Abbott, R. (1990). *Mad Mazes: Intriguing Mind Twisters for Puzzle Buffs, Game Nuts and Other Smart People*. Adams Media, 1990 (http://www.logicmazes.com).

Andre, D. & Russell, S. J. (2002). State Abstraction for Programmable Reinforcement Learning Agents. In *Proceedings of The 18th National Conference on Artificial Intelligence*.

Bernstein, D. S. (1999). *Reusing Old Policies to Accelerate Learning on New MDPs*. Technical Report 99–26, University of Massachusetts.

Blockeel, H. & De Raedt, L. (1998). Top–Down Induction of First Order Logical Decision Trees. *Artificial Intelligence* **101**(1–2): 285–297.

Bowling, M. & Veloso, M. (1998). Reusing Learned Policies Between Similar Problems. In *Proceedings of The AI*AI-98 Workshop on New Trends in Robotics*. Padua, Italy.

Bowling, M. & Veloso, M. (1999). Bounding the Suboptimality of Reusing Subproblems. In *Proceedings of The 16th International Joint Conference on AI*, 1340–1347. Sweden: Morgan-Kaufmann.

Breisemeister, L., Scheffer, T. & Wysotzki, F. (1995). *Combination of Problem Solving and Learning from Experience*. Technical Report 20/95, TU Berlin.

Breisemeister, L., Scheffer, T. & Wysotzki, F. (1996). A Concept Formation Based Algorithmic Model for Skill Acquisition. In *Proceedings of The First European Workshop on Cognitive Modelling*.

Carroll, J. L., Peterson, T. S. & Owen, N. E. (2001). Memory-guided Exploration in Reinforcement Learning. In *Proceedings of The International Joint Conference on Neural Networks*. Washington, DC.

Dietterich, T. (2000). Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research* **13**: 227–303.

Dixon, K. R., Malak, R. J. & Khosla, P. K. (2000). *Incorporating Prior Knowledge and Previously Learned Information into Reinforcement Learning Agents*. Technical Report, Institute for Complex Engineered Systems, Carnegie Mellon University.

Dreyfus, H. L. & Dreyfus, S. E. (1986). *Mind Over Machine: The Power of Human Intuition and Experience in the Era of the Computer*. Blackwell.

Džeroski, S., De Raedt, L. & Driessens, K. (2001). Relational Reinforcement Learning. *Machine Learning* **43**: 7–52.

Kaelbling, L. P., Littman, M. L. & Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research* **4**: 237–285.

Langley, P., Iba, W. & Thompson, K. (1992). An Analysis of Bayesian Classifiers. In *Proceedings of The 10th National Conference on Artificial Intelligence*, 223–228.

Maclin, R. & Shavlik, J. W. (1996). Creating Advice-Taking Reinforcement Learners. *Machine Learning* **22**: 251–282.

Michie, D., Bain, M. & Hayes-Michie, J. (1990). Cognitive Models from Subcognitive Skills. In McGhee, J., Grimble, M. J. & Mowforth, P. (eds.) *Knowledge-Based Systems for Industrial Control*, 71–99. Peter Peregrinus: London.

Ng, A. Y., Harada, D. & Russell, S. J. (1999). Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of The 16th International Conference on Machine Learning*.

Perkins, T. J. & Precup, D. (1999). *Using Options for Knowledge Transfer in Reinforcement Learning*. Technical Report 99–34, University of Massachusetts.

Russell, S. J. & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*, 2nd edn. Prentice Hall.

Šuc, D. (2001). *Skill Machine Reconstruction of Human Control Strategies*. Ph.D. dissertation, University of Ljubljana, Slovenia.

Quinlan, J. R. (1990). Learning Logical Definitions from Relations. *Machine Learning* **5**: 239–266.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Sun, R., Peterson, T. & Merrill, E. (1996). Bottom–up Skill Learning in Reactive Sequential Decision Tasks. In *Proceedings of The 18th Cognitive Science Society Conference*.

Sun, R. & Peterson, T. (1998). Autonomous Learning of Sequential Tasks: Experiments and Analyses. *IEEE Transactions on Neural Networks* **9**: 1217–1234.

Sun, R. & Merrill, E. (2001). From Implicit Skills to Explicit Knowledge: A Bottom–Up Model of Skill Learning. *Cognitive Science* **25**(2): 203–244.

Sutton, R. S. & Barto, A. S. (1998). *Reinforcement Learning*, *an Introduction*. MIT Press.

Sutton, R. S., McAllester, D. A., Singh, S. P. & Mansour, Y. (2000). Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Proceedings of Advances in Neural Information Systems*.

Tesauro, G. (1992). Practical Issues in Temporal Difference Learning. *Machine Learning* **8**: 257–277.

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

Thrun, S. & Schwartz, A. (1995). Finding Structure in Reinforcement Learning. In *Proceedings of Advances in Neural Information Systems*, 385–392.

Utgoff, P. E. & Cohen, P. R. (1998). Applicability of Reinforcement Learning. In *Proceedings of AAAI Workshop on The Methodology of Applying Machine Learning*.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. Ph.D. dissertation, Cambridge University.

Whitehead, S. D. (1991). *A Study of Cooperative Mechanisms for Faster Reinforcement Learning*. Technical Report 365, University of Rochester, New York.

Witten, I. H. & Frank, E. (2000). Data Mining*: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers.