Parallel Learning using Heterogeneous Agents

Patrick Mannion
Discipline of Information Technology
National University of Ireland Galway
p.mannion3@nuigalway.ie

Jim Duggan
Discipline of Information Technology
National University of Ireland Galway
jim.duggan@nuigalway.ie

Enda Howley
Discipline of Information Technology
National University of Ireland Galway
enda.howley@nuigalway.ie

ABSTRACT

Reinforcement Learning (RL) is a commonly used and effective Machine Learning technique, but can perform poorly when faced with complex problems leading to a slow rate of convergence. Parallel Learning (PL) is a novel paradigm within RL that seeks to address these concerns. In PL, multiple agents pool their experiences while learning concurrently on a problem, thus increasing performance and decreasing convergence times. Here we present a model-free Parallel Reinforcement Learning algorithm based on Q-Learning, which uses Heterogeneous Agents. Slave agents learn in parallel, where each learns on a different subset of the state action space for the given problem. The experience of these Slave agents is then transferred to a Master agent, where it is used for Q-Value initialisation. The Master agent then learns on the problem using the full state action space. Our approach is tested on several deterministic gridworld domains of varying sizes. We prove experimentally that our PL approach outperforms a standard Q-Learning agent, resulting in increased learning speed, and a lower average number of steps required to reach the goal state after the given training time. The benefits of our approach versus a standard Q-Learner are also found to be more significant as the number of pre-training episodes is increased.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentation

Keywords

Reinforcement Learning, Parallel Learning, Transfer Learning, Multi Agent Systems

1. INTRODUCTION

Reinforcement Learning (RL) is a commonly used and effective Machine Learning technique. RL algorithms are typically tested using abstract problem domains (e.g. gridworld, pole balancing, mountain car etc.), but numerous authors have also successfully applied RL to a range of more complex and realistic problem domains. These advanced RL application areas include cloud computing [1], urban traffic signal control [10], air traffic management [16], RoboCup Soccer [13], and video games [14].

However, many standard RL algorithms do not scale well when applied to complex environments with large state action spaces [5]. This shortcoming limits the potential of RL techniques when learning in challenging real world domains, resulting in reduced performance and high convergence times. Some more difficult problems may require hours or even days of training time to reach a useful solution. To increase the usefulness of RL when applied to complex problem domains like those mentioned above, it is essential that innovative techniques are developed to tackle these problems.

Parallel Learning (PL) is a novel paradigm within RL that is capable of addressing these concerns. In PL, multiple agents pool their experiences while learning concurrently on a problem, thus increasing performance and decreasing convergence times. In contrast to mainstream Multi Agent Reinforcement Learning (MARL) research, PL is typically applied to speeding up the convergence of single agent RL problems, rather than examining competitive or cooperative strategies and emergent behaviour as in MARL. PL agents typically influence each other's behaviour by sharing information, rather than the direct agent interactions that frequently occur in MARL. This is because PL agents generally learn in separate instances of the same problem, rather than multiple agents learning in a single problem instance as is common in MARL literature.

Here we present a novel variant of Parallel Reinforcement Learning, which uses Heterogeneous Agents. In our approach, multiple Slave agents with simplified action sets are allowed to learn on the same problem in parallel for a short pre-training period, and their combined experience is transferred to a Master agent which then learns on the full state action space. We test our approach against a standard Q-Learning agent on a variety of deterministic gridworld problems, and prove its efficacy with respect to learning speeds and quality of policy learned. These tests are repeated up to 1000 times to ensure that the results presented are both consistent and repeatable.

This paper will make a contribution in the following ways: 1) a novel Parallel Reinforcement Learning approach using Heterogeneous Agents is presented; 2) we prove experimentally that the proposed method improves learning speed while also reaching a better solution than a standard single learning agent approach in the training time given; 3) we analyse the effect of different amounts of pre-training on the performance of our algorithm across a range of problem dimensions; 4) we discuss the future direction of this research topic and the wider implications for RL research.

In the next section of this paper we examine previous related research, while in the third section we outline our Parallel Learning with Heterogeneous Agents approach. We then define our experimental procedure, which is followed by our experimental results. Our paper concludes with a discussion of our experimental findings and our plans to develop this work further in the future.

2. RELATED RESEARCH

2.1 Reinforcement Learning

Reinforcement Learning has become an increasingly popular Machine Learning technique in recent years, and has been applied successfully to a wide range of problem domains besides those mentioned above. Typically, an RL agent is situated in an environment, and learns how to behave through a process of reward and punishment. The agent usually has no prior knowledge about the environment into which it has been deployed. Each action selected by the learner receives a corresponding scalar reward signal from the environment, which depends on the previous and resultant environmental states. Actions that increase the utility of the system state are rewarded with a positive reward, while poor action choices would generally receive a negative reward. Thus, it is crucial to design an appropriate reward function for the application domain, as the goal of the RL agent is to maximise the reward received during its lifetime.

These problems are typically formulated as a Markov Decision Process (MDP), which has become the standard approach when defining problems that involve learning sequential decision making [18]. MDPs may be modelled using a set of states S, set of actions A, reward function R, and a transition function T [12]. Thus the full MDP is formulated as a tuple $< S, A, T, R >$. Selecting an action $a \in A$ in a given state s $\in$ S will result in the environment entering a new state $s' \in S$ with probability $T(s, a, s') \in$ (0,1), and give a reward $r = R(s, a, s')$.

There are two main categories of RL algorithms: model-based (e.g. Dyna, Prioritised Sweeping), and model-free (e.g. Q-Learning, SARSA). Model-based approaches require the transition function T to be known or learned for successful implementation [18], whereas this is not a requirement for model-free approaches. Model-free approaches instead sample the underlying MDP directly in order to gain knowledge about the unknown model.

In the model-free paradigm, the learner faces the problem of the exploration/exploitation dilemma, especially when it has no prior knowledge about its environment. A balance must be struck between exploration of new actions, and exploitation of known good actions. Too little exploration means that the agent may converge to a sub-optimal policy, while too much exploration means that the agent never gets to exploit the knowledge it has learned, resulting in poor performance. Several action selection strategies have been developed to manage this trade-off, including ϵ-greedy and Boltzmann (softmax).

Q-values represent the expected reward for selecting an action a while in a certain state s, and may be represented discretely in the form of a matrix, or by using a Q Function Approximation approach.

Among the numerous model-free approaches, two of the most commonly used include Q-Learning, and SARSA. Q-Learning is an off-policy learning algorithm which has been proven to converge to the optimum action-values with probability 1 so long as all actions are repeatedly sampled in all states and the action-values are represented discretely [17]. In Q-Learning, the Q values are updated according to Equation 1 below:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(r_t + \gamma \max_a(s_{t+1}, a) - Q_t(s_t, a_t)) \quad (1)$$

In the above equation, the learning rate $\alpha \in [0, 1]$ is an input parameter that determines the rate at which Q values are updated at each time step t. The discount factor $\gamma \in [0, 1]$ controls how the agent regards future rewards. A low value of γ results in an agent that is myopic, whereas a high value of γ means that the learner is more forward looking.

2.2 Parallel Reinforcement Learning

Kretchmar [7] describes a Parallel Reinforcement Learning (PRL) approach, whereby multiple agents learn in parallel on the same single agent RL task while sharing experience. Kretchmar tested his PRL implementation using a multi-armed bandit problem, with different numbers of agents learning in parallel. The results presented demonstrated that the PRL approach outperformed a single learner, and using additional parallel learners decreased the time taken to converge to the optimal control policy.

A comparative study of three different PRL implementations was published by Kushida et al. [8]. Two of the implementations tested were based on Q-Learning, while the third was based on fuzzy Q-Learning. The apporaches were tested on a gridworld problem, and the authors observed a clear benefit due to multiple agents learning in parallel compared to a single learner.

Grounds and Kudenko [4, 5] presented an approach which solves single agent RL tasks in parallel on a Beowulf cluster. They based their approach on SARSA, where learners share value function estimates which are represented by linear function approximators. By making use of the Message Passing Interface, agents asynchronously transmit messages containing their learned weight values. Experiments conducted by the authors proved that the time taken to compute good policies in single agent learning tasks can be reduced by applying PRL.

Li and Schuurmans [9] present several learning algorithms which are set up to execute in parallel and make use of the MapReduce framework. Parallel versions of dynamic programming, temporal difference and gradient temporal difference methods were presented.

Barrett et al. [2] developed a parallelised version of Q-Learning tailored for cloud resource allocation applications. This model-free algorithm was benchmarked against a single learning agent approach, and the authors found that the time taken to learn good policies was greatly reduced by parallelising the learning process.

A parallel framework for Bayesian Reinforcement Learning is described by Barrett et al. [1, 3]. In this model-based RL approach, agents learn state transition probabilities for a given problem domain without any prior knowledge. PRL agents learn in parallel on the same task, while sharing probability density estimates amongst each other. The goal of this approach is to speed up convergence. The authors test their approach using two different scenarios: a stochastic gridworld problem domain, and a cloud virtual machine al-

location problem which uses real world user demand data. The authors prove empirically that their proposed approach significantly improves convergence times in both test scenarios compared to a single agent learning on the same task, and use Kullback-Liebler Divergence as a performance metric. The benefits of the proposed PRL approach were also shown to scale well when additional PRL agents were added.

A Parallel Reinforcement Learning algorithm for Traffic Signal Control was proposed by Mannion et al. [11]. In this framework, multiple Q-Learning agents learn in parallel on separate instances of the same Traffic Signal Control problem while sharing experience, with the goal of improving learning speed and exploration. The authors tested their algorithm against a single learning agent on three Traffic Signal Control problems of varying complexity, and found that their PRL approach increased learning and exploration rates when compared to the single agent approach.

Despite promising results published by several authors using Parallel Reinforcement Learning approaches, surprisingly little research has been published in this area to date. In general, these new PRL approaches were tested using abstract problem domains, apart from Mannion et al. [11] and Barrett et al. [1, 2, 3], who also consider applications of PRL to complex real world problems.

3. PARALLEL REINFORCEMENT LEARNING USING HETEROGENEOUS AGENTS

In this section, we describe our Parallel Reinforcement Learning using Heterogeneous Agents (PRL-HA) approach. PRL-HA is unique compared to the approaches described above, in that it is proposed primarily as a method of learning experience to be transferred later to a single agent.

Typically in PRL literature, multiple agents are used to solve the same single agent RL problem. In PRL-HA, we aim to learn some approximate knowledge about the problem using multiple simple learners in parallel, and use this experience to initialise the Q-Values of another superior agent. Thus, our proposed approach could be considered to be a hybrid between Parallel Learning and Transfer Learning (TL). Research in Transfer Learning typically involves transferring learned experience between agents learning on different but related problems (see e.g. [15]), and has much in common with our proposed approach, in that the goal of both PRL-HA and TL is to improve the performance of a learner by initialising its Q-Values at the start of a task. In the case of PRL-HA, the experience transferred is collected by multiple inferior agents, for later use by a superior agent while learning on the same problem.

Our architecture consists of a single Master agent, and multiple Slave agents. These slave agents are diverse, in that each may only learn on a specific subset of the state action space. Both types of agents used in our architecture are based on Q-Learning.

Agents make their action selections using the ϵ-greedy strategy, where a random action is chosen with probability ϵ, or the action with the best expected reward is chosen with the remaining probability $1 - \epsilon$. For all agents in our experiments, the value of ϵ is set to 0.05. This value promotes exploitation of the knowledge the agent has gained, but still allows for some exploration.

We decompose the problem, by splitting the action set into multiple sub action sets, where each Slave agent learns using a different action set. The Slave agents are initially allowed to learn in parallel on the problem for a short duration, which we call the pre-training period. Each Slave agent learns on its own instance of the problem domain, and takes responsibility for acquiring knowledge about a specific subset of the state action space.

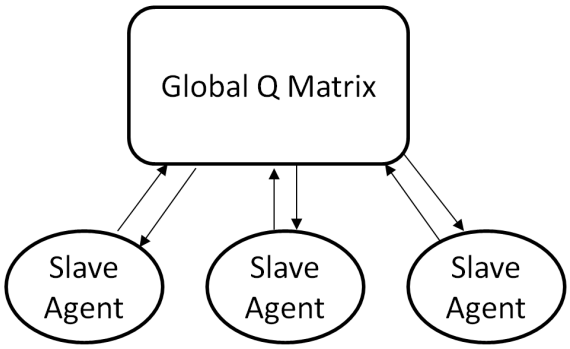

Figure 1: Multiple Slave Agents learn in parallel on the same problem, and their experience is combined into a Global Q Matrix

This initial approximate knowledge is transferred from the Slave agents into a global Q values matrix, as shown in Figure 1 above. Once the pre-training period is complete, the values stored in the global Q values matrix are transferred to the Master agent. Thus, PRL-HA may also be considered to be a form of Transfer Learning.

The Master agent then learns on the problem for the full duration using the complete action set, and may draw on the experience transferred from the simple Slave agents to aid its action selection choices. In effect, quite a lot of the required exploration has already been done for the Master agent when it begins learning on the problem.

The Slave agents are essentially incapable of solving the problem by themselves, e.g. for a 10 x 10 version of the gridworld problem presented in the next section, any one Slave agent has about a 9% chance of being able to reach the goal state in an episode. The chance of an individual Slave agent reaching the goal state in gridworld during a given episode becomes lower as the dimensionality of the problem increases.

The purpose of these Slave agents is not to solve the problem - rather they are used for very focused exploration. Each slave agent has the responsibility of exploring a specific subset of the state action space, with the goal being to get enough approximate knowledge during the pre-training period to benefit the Master agent when it begins to learn on the full state action space.

In the next section, we prove empirically that knowledge from these simple Slave agents can be combined into a coherent whole, and used successfully by a more complex Master agent which learns on the full state action space of the problem.

When applying PRL-HA to a problem, it is up to the designer to choose an appropriate method of sub-dividing the state action space, the choice of which will necessarily be highly domain specific. The subset of the state action space

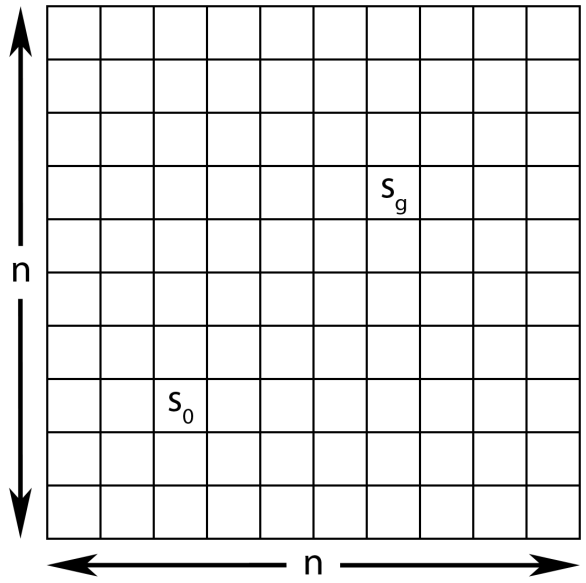

Figure 2: An n-dimensional gridworld problem, showing the start state (s_0) and goal state (s_g)

that each agent learns on needs to be defined using knowledge of the specific problem structure, as random division of portions of the state action space among Slave agents is unlikely to work well in all but the most simple problem domains. The requirement for domain specific knowledge from the system designer leads to interesting comparisons between our work and other areas of Reinforcement Learning research e.g. Reward Shaping. In Reward Shaping, the system designer guides the action selection choices of a single learning agent directly using domain specific knowledge (see e.g. Grzes and Kudenko [6]). By contrast, in PRL-HA domain specific knowledge is used in deciding which areas of the state action space will be assigned to individual Slave agents to explore.

4. EXPERIMENTAL DESIGN

As an initial empirical proof of the efficacy of our proposed approach, we have tested it on a series of experiments based on a deterministic gridworld. We focus here on demonstrating that our PRL approach can increase learning rates, while also improving the policy learned when compared to a standard Q-Learning agent.

Figure 2 shows a general case of the deterministic gridworld problem domain. The dimension of the gridworld n specifies the number of cells in the x and y dimensions. Each cell is considered to be a discrete environmental state, with the total size of the state space S equal to n^2. A state s is expressed as a vector containing the x and y coordinates of the cell in question, shown in Equation 2 below:

$$s = \left[\begin{array}{cc} x, & y \end{array} \right] \tag{2}$$

For the purposes of storing and retrieving Q values, a mixed radix conversion is used to represent a state vector as a single number. Each state receives a unique number under this scheme, which is used as a key for the relevant data in the Q values table.

An agent may only occupy a single state on the grid at any particular time. The agent starts at an initial state s_0, and must navigate its way to a goal state s_g. The gridworld episode ends once the goal state is reached. If the goal state has not been reached after 2000 moves, the current episode is ended and the gridworld is reset.

At the beginning of each episode, the start state s_0 is randomly selected. The location of the goal state is also randomly selected at the beginning of each experimental run. The agent is not allowed to start at the goal state, so s_0 is randomly selected until $s_0 \neq s_g$.

There are eight possible actions allowed in the gridworld: move North, North East, East, South East, South, South West, West, or North West. The complete action set A allowed in the gridworld is defined in Equation 3 below:

$$A = \{\ N,\ NE,\ E,\ SE,\ S,\ SW,\ W,\ NW\ \} \tag{3}$$

Each action chosen by an agent in the gridworld may result in one of two possible special outcomes: reaching the goal state, or moving off the grid. If an agent's action choice takes it to the goal state, a reward equal to n^2 is given and the gridworld episode is ended. If the resultant state is outside the gridworld, the agent is moved back to its previous position and given a reward of -2. All other possibilities apart from these special cases are given a reward of -1. Formally, the reward received by an agent for selecting action a in state s and transitioning to a resultant state s' is defined according to Equation 4 below:

$$R(s,a,s') = \begin{cases} n^2 & s' = s_g \\ -2 & s' \notin S \\ -1 & \text{otherwise} \end{cases} \tag{4}$$

To apply our PRL-HA approach to this problem domain, we use four Slave agents. Each one of these agents learns on a different subset of the state action space, with each having a limited action set. We define these action sets A_1, A_2, A_3 and A_4 formally in Equations 5 to 8 below. Each of the four Slave agents has a different set of two possible movement choices: North South, North-East South-West, East West, and South-East North-West.

$$A_1 = \{\ N,\ S\ \} \tag{5}$$

$$A_2 = \{\ E,\ W\ \} \tag{6}$$

$$A_3 = \{\ NE,\ SW\ \} \tag{7}$$

$$A_4 = \{\ SE,\ NW\ \} \tag{8}$$

The Slave agents are allowed to learn on the problem for a certain pre-training duration, and the combined experience of these Slave agents is transferred to the Master agent. The Master agent then learns on the full state action space, and is used to evaluate the effectiveness of our approach in this problem domain.

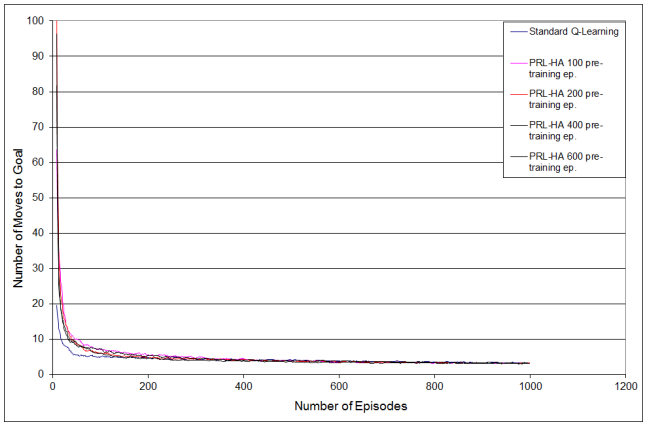

Figure 3: No. of Moves to Goal, 5x5 Gridworld

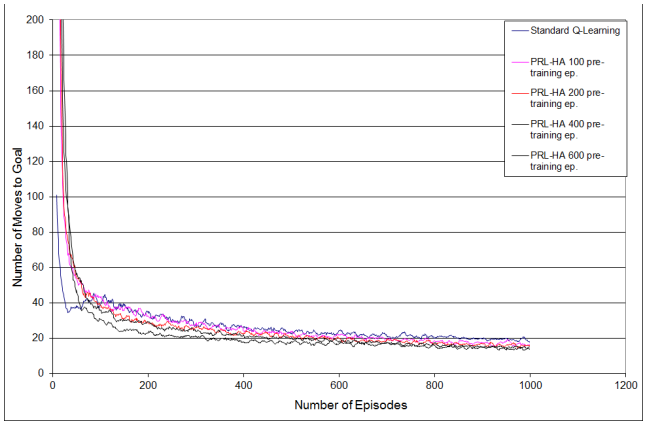

Figure 4: No. of Moves to Goal, 10x10 Gridworld

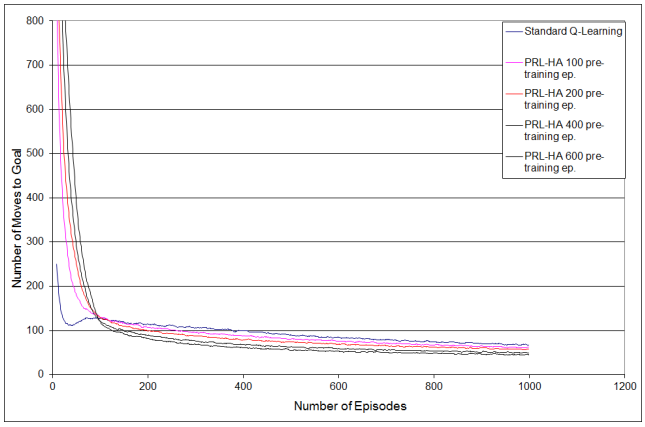

Figure 5: No. of Moves to Goal, 15x15 Gridworld

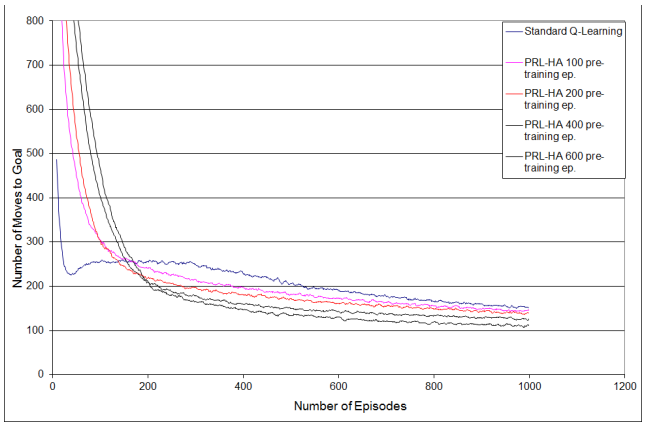

Figure 6: No. of Moves to Goal, 20x20 Gridworld

Our PRL-HA approach is evaluated using six different dimensions of the gridworld: $n = 5$, $n = 10$, $n = 15$, $n = 20$, $n = 25$, and $n = 30$. For each dimension of the gridworld, we test our PRL-HA approach against a single standard Q-Learning agent. This reference agent is the same in all respects as the Master agent mentioned above, except that it does not recieve any experience transfer, and its Q values are thus initialised to all zeroes.

We consider 4 different pre-training durations for the PRL-HA algorithm: 100, 200, 400 and 600 episodes. This gives a total of 30 experiments, which are each run for 1000 episodes. The results presented for $n = 5$ and $n = 10$ are the average of 100 runs. The multitude of possible locations of s_0 and s_g at $n \geq 15$ introduce extra variability into the performance of the agents, so we take the average of 1000 runs in these cases to ensure accuracy and repeatability. The learning rate α and the discount factor γ are set to 0.05 and 0.9 respectively for all agents used in our experiments. Table 1 summarises all of the values used in our experiments.

5. EXPERIMENTAL RESULTS AND DISCUSSION

Our experimental results are presented in Figures 3 to 8 and in Table 2 below. Figures 3 to 8 plot the number of moves taken to reach the goal state for the different gridworld dimensions tested. The lines plotted on these figures are the moving average of 10 points to improve clarity.

Table 2 summarises the results of all 30 experiments that were conducted, comparing the average number of moves to goal over the final 100 episodes for each test, as well as the percentage reduction in number of moves taken to reach the goal state when using PRL-HA compared to a standard Q-Learning agent.

Figure 9 plots the percentage reduction in moves to goal versus the gridworld dimension n for each pre-training duration considered. In general, it can be seen that our PRL-HA approach offers a clear advantage in terms of learning speed and quality of policy learned versus a standard Q-Learning agent, regardless of the number of pre-training episodes.

With respect to learning speeds, it is interesting to note the differences between the reference single agent and PRL-HA. In each gridworld test, we see that the reference agent initially converges more quickly than PRL-HA until they both reach a similar level of performance. After this point is reached, PRL-HA converges more quickly in almost all cases. The duration of this initial lag in the PRL-HA approach is typically of the order of 100 to 250 episodes, depending of course on the dimensionality of the problem and the amount

Table 1: Summary of Experimental Parameters

Parameter	Value(s) Used
Learning rate, α	0.05
Discount factor, γ	0.90
ϵ	0.10
Max. moves per episode	2000
Testing episodes	1000
Number of runs	100 ($n < 15$), 1000 ($n \geq 15$)
Pre-training episodes	100, 200, 400, 600
Gridworld dimension, n	5, 10, 15, 20, 25, 30

of pre-training allowed. We hypothesise that during this initial lag period the Master agent must assimilate the experience transferred to it into a coherent whole, and correct any contradictory information learned by the 4 slave agents. This lag period appears to occur consistently on all problem dimensions, and it would be interesting to analyse this phenomenon on a more challenging problem domain than gridworld in the future.

By considering the plot of percentage reduction in moves to goal versus the gridworld dimension n in Figure 9, we can gain some insight into the most appropriate pre-training durations to select for different gridworld sizes. At $n = 5$, there is a clear benefit to having either 100 or 200 episodes of pre-training, whereas 400 and 600 episodes are actually not as effective. However, this is a very simple problem (25 states in total), so it is obvious in this case that too much pre-training may actually be detrimental to the learner's performance, resulting in more incorrect information being learned compared to 200 episodes of pre-training. Even so, every duration of pre-training tested for the $n = 5$ case still produced better results than the reference agent.

We begin to see more significant results at $n = 10$, which has 4 times as many states as $n = 5$. Here the benefits of PRL-HA begin to really show, with each increase in pre-training time delivering a corresponding decrease in average number of moves to goal. This trend continues for the other values of n, and we can say that in general, more pre-training time leads to a faster rate of learning and a better final learned policy in our experiments. During our experimentation, we found that setting the number of pre-training episodes to 10 times n allowed a significant boost in performance (of the order of 8 to 10%) across all gridworld sizes tested, without spending an excessively long time on pre-training. Also, similar to the case with $n = 5$, there may be an amount of pre-training which will produce the best results for other problem domains, beyond which extra pre-training would be detrimental to performance.

Our PRL-HA approach reduced the average number of moves taken to s_g in practically all cases, regardless of the amount of pre-training given. The only exception to this was on the $n = 30$ gridworld test, where PRL-HA with 100 episodes of pre-training took 0.55% more steps on average to reach the goal state when compared with the reference single learning agent.

This is not a very significant decline in performance, and can be attributed to the fact that 100 episodes of pre-training do not provide sufficient information for PRL-HA to show any benefit when applied to the most complicated domain used in our tests. However, doubling the number of pre-training episodes to 200 for this value of n means that PRL-HA performs nearly 4% better than the reference agent. This highlights the fact that additional pre-training episodes may be required in more complex problem domains for the benefits of PRL-HA to become apparent.

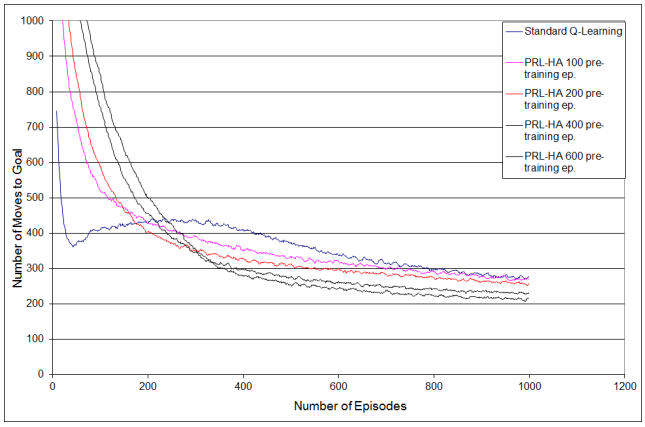

Figure 7: No. of Moves to Goal, 25x25 Gridworld

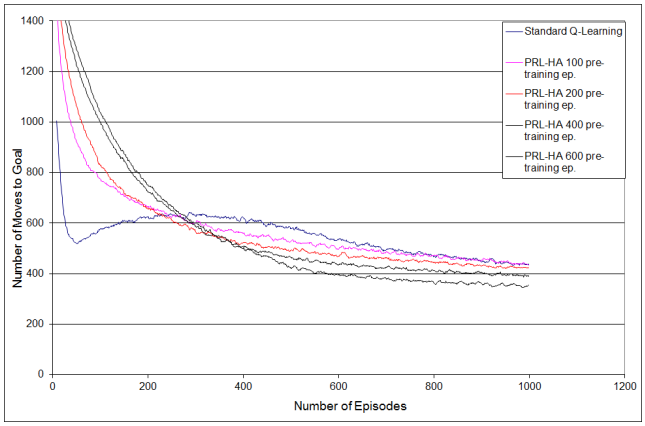

Figure 8: No. of Moves to Goal, 30x30 Gridworld

6. CONCLUSIONS AND FUTURE WORK

In this paper we have described a novel variant of Parallel Reinforcement Learning which makes use of Heterogeneous Agents. Slave agents learn on a subset of the state action space for a short pre-training period, after which their experience is transferred to a Master agent. Using a comprehensive suite of tests based on progressively more complex gridworld problems, we have proven empirically that PRL-HA consistently outperforms a standard single Q-Learning agent, both in terms of learning rates and performance. The results presented were averaged over up to 1000 runs to ensure their significance, accuracy and repeatability.

We have also investigated the effects of different amounts of pre-training on PRL-HA, showing that even a low number of pre-training episodes is generally sufficient to give an improvement in performance versus a standard single agent learner. Furthermore, we demonstrated that higher gains can be achieved by increasing the number of pre-training

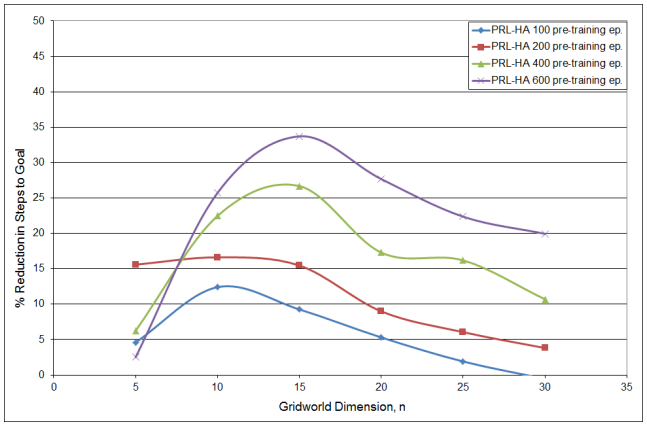

Figure 9: Percentage Reduction in No. of Moves to Goal vs. Gridworld Dimension n (Final 100 Episodes)

episodes, especially when this approach is applied to more complex problems. Typically, improvements of up to 30% in number of moves to goal are possible using PRL-HA, depending on the problem dimensionality and the number of pre-training episodes used. Our results demonstrate that preliminary exploration of an environment with a set of inferior heterogeneous agents may be useful when their knowledge is transferred to a superior agent learning in the same problem domain.

The approach that we have presented is based on Q-Learning, but the basic principles of PRL-HA should also be effective when used with other Reinforcement Learning algorithms, both model-based and model-free. Accordingly, we intend to test this method further by substituting different RL algorithms in place of Q-Learning. It would also be interesting to investigate the effect of using function approximation with PRL-HA in place of full state representation as used in this paper.

While we have tested PRL-HA on a simple deterministic gridworld problem here as a proof of concept, in future we plan to test our approach on more complex problem domains that are not deterministic. Advanced application areas for PRL-HA that we are currently investigating include urban traffic signal control and cloud resource allocation problems. As we discussed earlier, sub-dividing the state action space among Slave agents is an essential part of this algorithm. The division of the state action space is straightforward in the gridworld problems used in this paper, but this will not be the case when moving to more complex problem domains. Thus, one of the main difficulties of applying PRL-HA to these difficult problems is deciding how best to sub-divide the state action space, and it is unlikely that simple approaches like random allocation of action subsets to Slave agents will be effective. Rather, a carefully considered division of the state action space will have to be developed for each application area. We expect that PRL-HA will be effective when applied to problems with large state action spaces once appropriate state action space sub-divisions are discovered.

Another factor that could be explored in the future is the effect of varying the numbers of Slave agents used for pre-training. In all of the experiments in this paper, we used 4 Slave agents during the pre-training phase. For any application domain, there may be an optimum number of Slave agents to use, or an optimum number of divisions to make in the state action space, and this will need to be considered in future work on this topic.

Table 2: Summary of Experimental Results (Average over Final 100 Episodes)

Exp.	n	Pre-Training	No. Moves	% Reduction
1	5	-	3.40	-
2	5	100 ep.	3.24	4.60 %
3	5	200 ep.	2.87	15.15 %
4	5	400 ep.	3.19	6.26 %
5	5	600 ep.	3.31	2.51 %
6	10	-	19.29	-
7	10	100 ep.	16.90	12.41 %
8	10	200 ep.	16.08	16.62 %
9	10	400 ep.	14.96	22.46 %
10	10	600 ep.	14.33	25.72 %
11	15	-	68.93	-
12	15	100 ep.	62.54	9.26 %
13	15	200 ep.	58.27	15.47 %
14	15	400 ep.	50.53	26.68 %
15	15	600 ep.	45.69	33.71 %
16	20	-	154.67	-
17	20	100 ep.	146.47	5.30 %
18	20	200 ep.	140.77	8.99 %
19	20	400 ep.	127.93	17.29 %
20	20	600 ep.	111.86	27.68 %
21	25	-	277.42	-
22	25	100 ep.	272.12	1.91 %
23	25	200 ep.	260.62	6.06 %
24	25	400 ep.	232.50	16.19 %
25	25	600 ep.	215.37	22.37 %
26	30	-	442.56	-
27	30	100 ep.	445.02	-0.55 %
28	30	200 ep.	425.73	3.81 %
29	30	400 ep.	395.25	10.69 %
30	30	600 ep.	354.33	19.94 %

7. ACKNOWLEDGMENTS

The primary author would like to acknowledge the financial support provided to him by the Irish Research Council, through the Government of Ireland Postgraduate Scholarship Scheme.

8. REFERENCES

[1] E. Barrett, J. Duggan, and E. Howley. A parallel framework for bayesian reinforcement learning. *Connection Science*, 26(1):7–23, Jan. 2014.

[2] E. Barrett, E. Howley, and J. Duggan. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and Computation: Practice and Experience*, 25(12):1656–1674, 2013.

[3] E. Barrett, E. Howley, and J. Duggan. Parallel bayesian model learning. In *Proceedings of the*

Adaptive and Learning Agents workshop (at AAMAS 2013), May 2013.

[4] M. Grounds and D. Kudenko. Parallel reinforcement learning with linear function approximation. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, pages 45:1–45:3, New York, NY, USA, 2007. ACM.

[5] M. Grounds and D. Kudenko. Parallel reinforcement learning with linear function approximation. In K. Tuyls, A. Nowe, Z. Guessoum, and D. Kudenko, editors, *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, volume 4865 of *Lecture Notes in Computer Science*, pages 60–74. Springer Berlin Heidelberg, 2008.

[6] M. Grzes and D. Kudenko. Learning potential for reward shaping in reinforcement learning with tile coding. In *Proceedings of the Adaptive Learning Agents and Multi-Agent Systems workshop (at AAMAS 2008)*, May 2008.

[7] R. M. Kretchmar. Parallel reinforcement learning. In *The 6th World Conference on Systemics, Cybernetics,and Informatics, Orlando, FL.*, 2002.

[8] M. Kushida, K. Takahashi, H. Ueda, and T. Miyahara. A comparative study of parallel reinforcement learning methods with a pc cluster system. In *Intelligent Agent Technology, 2006. IAT '06. IEEE/WIC/ACM International Conference on*, pages 416–419, Dec 2006.

[9] Y. Li and D. Schuurmans. Mapreduce for parallel reinforcement learning. In S. Sanner and M. Hutter, editors, *Recent Advances in Reinforcement Learning*, volume 7188 of *Lecture Notes in Computer Science*, pages 309–320. Springer Berlin Heidelberg, 2012.

[10] P. Mannion, J. Duggan, and E. Howley. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In A. Kotsialos, F. Kluegl, L. McCluskey, J. P. Mueller, O. Rana, and R. Schumann, editors, *Autonomic Road Transport Support Systems*, Autonomic Systems. Birkhauser/Springer, 2015 (in press).

[11] P. Mannion, J. Duggan, and E. Howley. Parallel reinforcement learning for traffic signal control. In *Proceedings of the 4th International Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications (ABMTRANS 2015)*, June 2015 (in press).

[12] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[13] P. Stone, G. Kuhlmann, M. E. Taylor, and Y. Liu. Keepaway soccer: From machine learning testbed to benchmark. In I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, editors, *RoboCup-2005: Robot Soccer World Cup IX*, volume 4020, pages 93–105. Springer-Verlag, Berlin, 2006.

[14] M. E. Taylor, N. Carboni, A. Fachantidis, I. Vlahavas, and L. Torrey. Reinforcement learning agents providing advice in complex video games. *Connection Science*, 26(1):45–63, 2014.

[15] M. E. Taylor and P. Stone. Behavior transfer for value-function-based reinforcement learning. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, and M. Wooldridge, editors, *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 53–59, New York, NY, July 2005. ACM Press.

[16] K. Tumer and A. Agogino. Distributed agent-based air traffic flow management. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 330–337, Honolulu, HI, May 2007.

[17] C. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[18] M. Wiering and M. van Otterlo, editors. *Reinforcement Learning: State-of-the-Art*. Springer, 2012.