Concurrent PAC RL

Zhaohan Guo and **Emma Brunskill**
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh PA, 15213
United States

Abstract

In many real-world situations a decision maker may make decisions across many separate reinforcement learning tasks in parallel, yet there has been very little work on concurrent RL. Building on the efficient exploration RL literature, we introduce two new concurrent RL algorithms and bound their sample complexity. We show that under some mild conditions, both when the agent is known to be acting in many copies of the same MDP, and when they are not the same but are taken from a finite set, we can gain linear improvements in the sample complexity over not sharing information. This is quite exciting as a linear speedup is the most one might hope to gain. Our preliminary experiments confirm this result and show empirical benefits.

The ability to share information across tasks to speed learning is a critical aspect of intelligence, and an important goal for autonomous agents. These tasks may themselves involve a sequence of stochastic decisions: consider an online store interacting with many potential customers, or a doctor treating many diabetes patients, or tutoring software teaching algebra to a classroom of students. Here each task (customer relationship management, patient treatment, student tutoring) can be modeled as a reinforcement learning (RL) problem, with one decision maker performing many tasks in parallel. In such cases there is an opportunity to improve outcomes for all tasks (customers, patients, students) by leveraging shared information across the tasks.

Interestingly, despite these compelling applications, there has been almost no work done on concurrent reinforcement learning. There has been a number of papers (e.g. (Evgeniou and Pontil 2004; Xue et al. 2007)) on supervised concurrent learning (referred to as multi-task learning). In this context, multiple supervised learning tasks, such as classification, are run in parallel, and information from each is used to speed learning. When the tasks themselves involve sequential decision making, like reinforcement learning, prior work has focused on sharing information serially across consecutive related tasks, such as in transfer learning (e.g. (Taylor and Stone 2009; Lazaric and Restelli 2011)) or online learning across a set of tasks (Brunskill and Li 2013). Note that multiagent literature considers multiple agents acting in a single environment, whereas we consider the different problem of one agent / decision maker simultaneously acting in multiple environments. The critical distinction here is that the actions and rewards taken in one task do not directly impact the actions and rewards taken in any other task (unlike multiagent settings) but information about the outcomes of these actions may provide useful information to other tasks, if the tasks are related.

One important exception is recent work by Silver et al. (2013) on concurrent reinforcement learning when interacting with a set of customers in parallel. This work nicely demonstrates the substantial benefit to be had by leveraging information across related tasks while acting jointly in these tasks, using a simulator built from hundreds of thousands of customer records. However, this paper focused on an algorithmic and empirical contribution, and did not provide any formal analysis of the potential benefits of concurrent RL in terms of speeding up learning.

Towards a deeper understanding of concurrent RL and the potential advantages of sharing information when acting in parallel, we present new algorithms for concurrent reinforcement learning and provide a formal analysis of their properties. More precisely, we draw upon literature on Probably Approximately Correct (PAC) RL (Kearns and Singh 2002; Brafman and Tennenholtz 2002; Kakade 2003), and bound the sample complexity of our approaches, which is the number of steps on which the agent may make a sub-optimal decision, with high probability. Interestingly, when all tasks are identical, we prove that simply by applying an existing state-of-the-art single task PAC RL algorithm, MBIE (Strehl and Littman 2008), we can obtain, under mild conditions, a linear improvement in the sample complexity, compared to learning in each task with no shared information. We next consider a much more generic situation, in which the presented tasks are sampled from a finite, but unknown number of discrete state–action MDPs, and the identity of each task is unknown. Such scenarios can arise for many applications in which an agent is interacting with a group of people in parallel: for example, Lewis (Lewis 2005) found that when constructing customer pricing policies for news delivery, customers were best modeled as being one of two (latent) types, each with distinct MDP parameters. We present a new algorithm for this setting and prove that under fairly general conditions, that if any two distinct MDPs differ in their model parameters by a minimum gap for at least one state–action pair, and the MDPs have finite diameter, that we can also obtain essentially a linear improvement in the sample complexity bounds across identical tasks. Our approach incurs no dominant overhead in sample complexity by having to perform a clustering amongst tasks, implying that if all tasks are distinct, the resulting (theoretical) performance will be equivalent to as if we performed single task PAC RL in each task separately. These results provide an interesting counterpart to the sequential transfer work of Brunskill and Li (2013) which demonstrated a reduction in sample complexity was possible if an agent performed a series of tasks drawn from a finite set of MDPs; however, in contrast to that work that could only gain a benefit after completing many tasks, clustering them, and using that knowledge for learning in later tasks, we demonstrate that we can effectively cluster tasks and leverage this clustering during the reinforcement learning of those tasks to improve performance. We also provide small simulation experiments that support our theoretical results and demonstrate the advantage of carefully sharing information during concurrent reinforcement learning.

Background

A Markov decision process (MDP) is a tuple $\langle S, A, T, R, \gamma \rangle$ where S is a set of states, A is a set of actions, T is a transition model where $T(s'|s,a)$ is the probability of starting in state s, taking action a and transitioning to state s', $R(s,a) \in [0,1]$ is the expected reward received in state s upon taking action a, and γ is an (optional) discount factor. When it is clear from context we may use S and A to denote $|S|$ and $|A|$ respectively. A policy π is a mapping from states to actions. The value $V^{\pi}(s)$ of a policy π is the expected sum of discounted rewards obtained by following π starting in state s. We may use $V(s)$ when the policy is clear in context. The optimal policy π^* for a MDP is the one with the highest value function, denoted $V^*(s)$.

In reinforcement learning (RL) the transition and reward models are initially unknown. Probably Approximately Correct (PAC) RL methods (Kearns and Singh 2002; Brafman and Tennenholtz 2002; Strehl, Li, and Littman 2006) guarantee the number of steps on which the agent will make a less than ϵ-optimal decision, the sample complexity, is bounded by a polynomial function of the problems' parameters, with high probability. Sample complexity can be viewed as a measure of the learning speed of an algorithm, since it bounds the number of possible mistakes the algorithm will make. We will similarly use sample complexity to formally bound the potential speedup in learning gained by sharing experience across tasks.

Our work builds on MBIE, a single-task PAC RL algorithm (Strehl and Littman 2008). In MBIE the agent uses its experience to construct confidence intervals over its estimated transition and reward parameters. It computes a policy by performing repeated Bellman backups which are optimistic with respect to its confidence intervals, thereby constructing an optimistic MDP model, an optimistic estimate of the value function, and an optimistic policy. This policy will drive the agent towards little experienced state–action pairs or state–action pairs with high reward. We chose to build on MBIE due to its good sample complexity bounds and very good empirical performance.

We think it will be similarly possible to create concurrent algorithms and analysis building on other single-agent RL algorithms with strong performance guarantees, such as recent work by Lattimore, Hutter and Sunehag (2013), but leave this direction for future work.

Concurrent RL in Identical Environments

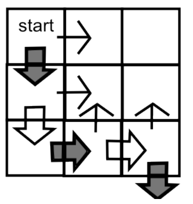
We first consider a decision maker (a.k.a agent) performing concurrent RL across a set of K MDP tasks. The model parameters of the MDP are unknown, but the agent does know that all K tasks are the same MDP. At time step t, each MDP k is in a particular state s_{tk}. The decision maker then specifies an action for each MDP $a_1, \ldots, a_K$. The next state of each MDP then is generated given the stochastic dynamics model $T(s'|s,a)$ for the MDP and all the MDPs synchronously transition to their next state. This means the actual state (and reward) in each task at each time step will typically differ. [1]. In addition there is no interaction between the tasks: imagine an agent coordinating the repair of many identical-make cars. Then the state of repair in one car does not impact the state of repair of another car.

We are interested in formally analyzing how sharing all information can impact learning speed. At best one might hope to gain a speedup in learning that scales exactly linearly with the number of MDPs K. Unfortunately such a speedup is not possible in all circumstances, due to the possibility of redundant exploration. For example, consider a small MDP where all the MDPs start in the same initial state. One action transitions to a part of the state space with low rewards, and another action to a part with high rewards. It takes a small number of tries of the bad action to learn that it is bad. However in the concurrent setting, if there are many many MDPs, then the bad action will be tried much more than necessary because the rest of the states have not yet been explored. This potential redundant exploration is inherently due to the concurrent, synchronous, online nature of the problem, since the decision maker must assign an action to each MDP at each time step, and can't wait to see the outcomes of some decisions before assigning other actions to other MDPs.

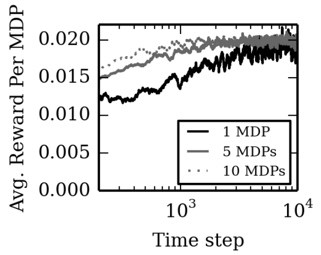
Interestingly, we now show that a trivial extension of the MBIE algorithm is sufficient to achieve a linear improvement in the sample complexity for a very wide range of K, with no complicated mechanism needed to coordinate the exploration across the MDPs. Our concurrent MBIE (CMBIE) algorithm uses the MBIE algorithm in its original form except we share the experience from all K agents.

We now give a high-probability bound on the total sample complexity across all K MDPs. As at each time step the algorithm selects K actions, our sample complexity is a bound on the total number of non-ϵ-optimal actions selected (not just the number of steps). Proofs, when omitted

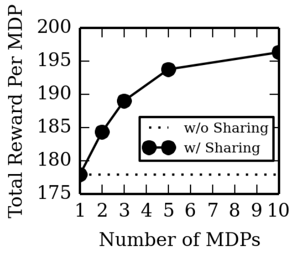
[1]We suspect it will be feasible to extend to asynchronous situations but for clarity we focus on synchronous execution and leave asynchronous actions for future work.



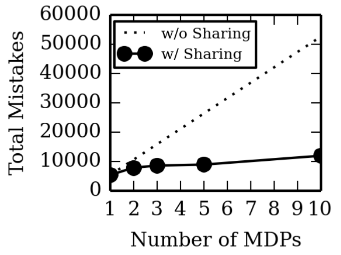
(a) Skinny/ filled thick/ empty thick arrows yield reward 0.03/ 0.02/ 1 with prob 1/ 1/ 0.02.



(b) Average reward per MDP per time step for CMBIE, when running in 1, 5, or 10 copies of the same MDP. A sliding window average of 100 steps is used for readability.



(c) Total cumulative reward per MDP after 10000 time steps versus number of MDPs.



(d) The number of total mistakes made after 10000 time steps versus number of MDPs.

Figure 1: CMBIE Experiments

are available in the accompanying tech report.[2]

Theorem 1. *Given ϵ and δ, and K agents acting in identical copies of the same MDP, Concurrent MBIE (CMBIE) will select an ϵ-optimal action for all K agents on all but at most*

$$\tilde{O}\left(\frac{1}{\epsilon(1-\gamma)^2}\left[\frac{S^2A}{\epsilon^2(1-\gamma)^4}+SA(K-1)\right]\right) \quad (1)$$

actions, with probability at least $1-\delta$, where $\tilde{O}$ neglects multiplicative log factors.

We use Strehl, Li and Littman (2006)'s generic PAC-MDP theorem to analyze the sample complexity of our approach.

An alternative is the agents share no information. Using the MBIE bound for a single MDP (Strehl and Littman 2008), $\tilde{O}\left(\frac{S^2A}{\epsilon^3(1-\gamma)^6}\right)$, no-sharing MBIE yields a sample complexity of $\tilde{O}\left(\frac{S^2AK}{\epsilon^3(1-\gamma)^6}\right)$. Taking the ratio of this with our CMBIE sample complexity bound yields potential improvement factor of $\tilde{O}\left(\frac{SK\frac{1}{\epsilon^2(1-\gamma)^4}}{\left[\frac{S}{\epsilon^2(1-\gamma)^4}+K\right]}\right)$. We now consider how this scales as a function of K. If $K \gg \frac{S}{\epsilon^2(1-\gamma)^4}$, we will obtain an approximately constant speedup that does not increase further with more concurrent agents. However, if $K \leq \frac{S}{\epsilon^2(1-\gamma)^4}$, the speedup is approximately K. This suggests that until K becomes very large, we will gain a linear improvement in the sample complexity as more agents are added.

This is quite encouraging, as it implies that by performing concurrent RL across a set of identical MDPs, we expect to get a linear speedup in the sample complexity as a function of the number of concurrent MDP copies/agents compared to not sharing information.

CMBIE Experiments

Prior work by Silver et al. (2013) has already nicely demonstrated the potential empirical benefits of concurrent reinforcement learning for a large customer marketing simulation. Our primary contribution is a theoretical analysis of the potential benefits of concurrent RL for efficient RL. However, sample complexity bounds are known to be quite conservative. We now illustrate that the benefits suggested by our theoretical results can also lead to empirical improvements in a small simulation domain.

We use a 3x3 gridworld (Figure 1(a)). The agent moves in the 4 cardinal directions deterministically. Hitting a wall results in no change, except moving down in the bottom-right state will transition to the top-left start state. The arrows display different reward dynamics for those actions. The optimal path to take is along the thick arrows, which will give expected reward of 0.02 per step.

Silver et al. obtained encouraging empirical rewards by using an ϵ-greedy style approach for concurrent RL in identical continuous-state environments, but we found that MBIE performed much better than ϵ-greedy for our discrete state–action MDP, so we focused on MBIE.

Each run was for 10000 time steps and all experiments were averaged over 100 runs. As is standard, we treat the size of the confidence sets over the reward and transition parameter estimates (given the experience so far) as tunable parameters. We tuned the confidence interval parameters to maximize the cumulative reward for acting in a single task, and then used the same settings for all concurrent RL scenarios. We set $m = \infty$, which essentially corresponds to always continuing to improve and refine the parameter estimates (fixing them after a certain number of experiences is important for the theoretical results but empirically it is often best to use all available experience).

Figure 1(b) shows that CMBIE achieves a significant increase in how fast it learns the best policy. Figure 1(c) also shows a significant gain in total rewards as more sharing is introduced. A more direct measure of sample complexity is to evaluate the number of mistakes (when the agent does not follow an ϵ-optimal policy) made as the agent acts. CMBIE incurs a very small cost from 1-10 agents, significantly better than if there is no sharing of information (Figure 1(d)), as indicated by the dotted line. These results provide preliminary empirical support that concurrent sample efficient RL demonstrates the performance suggested by our theoretical results, and also results in higher rewards in practice.

[2]The tech report is available at http://www.cs.cmu.edu/~ebrun/publications.html

Concurrent RL in Different Environments

Up to this point we have assumed that the agent is interacting with K MDPs, and it knows that all of them are identical. We now consider a more general situation where the K MDPs are drawn from a set of N distinct MDPs (with the same state–action space), and the agent does not know in advance how many distinct MDPs there are, nor does it know the identity of each MDP (in other words, it does not know in advance how to partition K into clusters where all MDPs in the same cluster have identical parameters). As an example, consider a computer game being played in parallel by many users. We may cluster the users as being 1 of N different types of how they may overcome the challenges in the game.

We propose a two-phase algorithm (ClusterCMBIE) to tackle this setting:

1. Run PAC-EXPLORE (Algorithm 1) in each MDP.
2. Cluster the MDPs into groups of identical MDPs.
3. Run CMBIE for each cluster.

For the first phase, we introduce a new algorithm, PAC-EXPLORE. The sole goal of PAC-EXPLORE is to explore the state–action pairs in each MDP sufficiently well so that the MDPs can be accurately clustered together with other MDPs that have identical model parameters. It does not try to act optimally, and is similar to executing the explore policy in E^3 (Kearns and Singh 2002), though we use confidence intervals to compute the policy which works better in practice. Specifically, PAC-EXPLORE takes input parameters m_e and T, and will visit all state–action pairs in an MDP at least m_e times. PAC-EXPLORE proceeds by dividing the state–action space into those pairs which have been visited at least m_e times (the known pairs) and those that have not. It constructs a MDP $\widehat{M}_K$ in which for known pairs, the reward is 0 and the transition probabilities are the estimated confidence sets (as in MBIE), and for the unknown pairs the reward is 1 and the transitions are deterministic self loops. It then constructs an optimistic (with respect to the confidence sets) T-step policy for $\widehat{M}_K$ which will tend to drive towards visiting unknown pairs. This T-step policy is repeated until an unknown pair is visited, then which the least tried action is taken (balanced wandering), which is then repeated until a known pair is visited, at which point a new optimistic T-step policy is computed and this procedure repeats. The use of episodes was motivated by our theoretical analysis, and has the additional empirical benefit of reducing the computational burden of continuous replanning.

Once phase 1 finishes, we compute confidence intervals over the estimated MDP model parameters for all state–action pairs for all MDPs. For any two MDPs, we place the two in the same cluster if and only if their confidence intervals overlap for all state–action pairs. The clustering algorithm proceeds by comparing the first MDP with all the other MDPs, pooling together the ones that don't differ from the first MDP. This creates the first cluster. This procedure is then repeated until all MDPs are clustered (which may create a cluster of cardinality one). This results in at most $\binom{N}{2} \leq N^2$ checks.

We now show our approach can yield a substantially lower sample complexity compared to not leveraging shared experience. Our analysis relies on two quite mild assumptions:

1. Any two distinct MDPs must differ in their model parameters for at least one state–action pair by a gap Γ (e.g. the L1 distance between the vector of their parameters for this state–action pair must be at least Γ).
2. All MDPs must have a finite diameter (Jaksch, Ortner, and Auer 2010) (denoted by D).

This is a very flexible setup, and we believe our gap condition is quite weak. If the dynamics between the distinct MDPs had no definite gap, we would incur little loss from treating them as the same MDP. However, in order for our algorithm to provide a benefit over a no sharing approach, Γ must be larger than the $\epsilon(1-\gamma)$ accuracy in the model parameter estimates that is typically required in single task PAC approaches (e.g. Strehl, Li and Littman (2006)). Intuitively this means that it is possible to accurately cluster the MDPs before we have sufficient experience to uncover an ϵ-optimal policy for a MDP. Our second assumption of a finite diameter is to ensure we can explore the MDP without getting stuck in a subset of the state–action space. The diameter of an MDP is the expected number of steps to go between any two states under some policy.

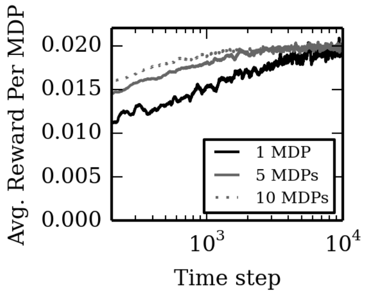
We first present a few supporting lemma, before providing a bound on how long phase 1 will take using our PAC-EXPLORE algorithm. We then use this to provide a bound on the resulting sample complexity. For the lemmas, let M be an MDP. Let $\widehat{M}$ be a generalized, approximate MDP with the same state–action space and reward function as M but whose transition functions are confidence sets, and each action also includes picking a particular transition function for the state at the current timestep.

Lemma 1. *Generalized Undiscounted Simulation Lemma*
Suppose the transition confidence sets of $\widehat{M}$ are ϵ-approximations to M (i.e. any possible transition function is within ϵ in L1 distance). Then for all T-step policies π, states s, and times $t < T$ we have that $|V_{\pi,t,\widehat{M}}(s) - V_{\pi,t,M}(s)| < \epsilon T^2$ where $V_{\pi,t,M}(s)$ is the expected undiscounted cumulative reward from time t to T when following π in M.

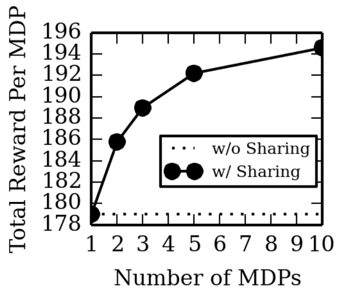
Lemma 2. *Optimistic Undiscounted Values*
Let $P_T(s,a)$ be the confidence set of transition probabilities for state–action (s,a) in $\widehat{M}$, and assume they contain the true transition probabilities of M. Then performing undiscounted value iteration with the update rule $Q_{t,T,\widehat{M}}(s,a) = R(s,a) + \max_{T' \in P_T}\left(\sum_{s'} T'(s'|s,a)V_{t+1,T,\widehat{M}}\right)$ results in an optimistic q–value function i.e. $Q_{t,T,\widehat{M}}(s,a) \geq Q^*_{t,T,M}(s,a)$ for all state–actions (s,a) and time steps $t \leq T$.

Theorem 2. *PAC-EXPLORE will visit all state–action pairs at least m_e times in no more than $\tilde{O}\left(SADm_e\right)$ steps, with probability at least $1-\delta$, where $m_e = \tilde{\Omega}\left(SD^2\right)$.*

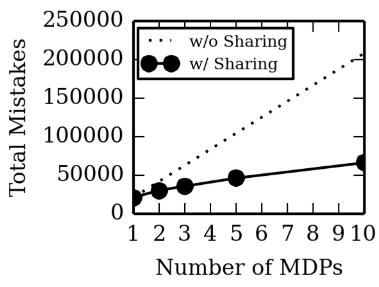
Proof. Consider the very start of a new episode. Let K be the set of state–action pairs that have been visited at least



(a) Average reward per MDP per time step for clustering CMBIE, when running in 1, 5, or 10 copies of each MDP type. A sliding window average of 100 steps is used for readability.



(b) Total cumulative reward per MDP after 10000 time steps versus number of MDPs.



(c) The number of total mistakes made after 10000 time steps versus number of MDPs.

Figure 2: ClusterCMBIE Experiments

$m_e = \Omega(S \log(S/\delta)/\alpha^2))$ times, the known pairs, where α will later be specified such that $m_e = \tilde{\Omega}(SD^2)$. Then the confidence intervals around the model parameter estimates will be small enough such that they are α-approximations (in L1 distance) to the true parameter values (by Lemma 8.5.5 in Kakade (2003)). Let M_K be the same as $\widehat{M}_K$ except the transition model for all known state–action pairs are set to their true parameter values. Since the diameter is at most D, that means in the true MDP M, there exists a policy π that takes at most expected D steps to reach an unknown state (escape). By Markov's inequality, we know there is a probability of at least $(c-1)/c$ that π will escape within cD steps and obtain a reward of dD in $(c+d)D$ steps in M_K. Then the expected value of this policy in a $T = (c+d)D$ length episode is at least $\frac{(c-1)dD}{c}$. Now we can compute an optimistic T-step policy $\widehat{\pi}$ in $\widehat{M}_K$. Then $\widehat{\pi}$'s expected value in $\widehat{M}_K$ is also at least $\frac{(c-1)dD}{c}$ (Lemma 2). Applying Lemma 1, $\widehat{\pi}$'s expected value in M_K, which can be expressed as $\sum_{t=1}^{T} \Pr(\text{escapes at } t)(T-t)$, is at least $\frac{(c-1)dD}{c} - \alpha((c+d)D)^2$. Then the probability of escaping at any step in this episode, p_e is at least

$$\begin{aligned} p_e = \sum_{t=1}^{T} \Pr(\text{escapes at } t) &\geq \sum_{t=1}^{T} \Pr(\text{escapes at } t)\frac{(T-t)}{T} \\ &\geq \frac{(c-1)dD}{cT} - \frac{\alpha((c+d)D)^2}{T} \\ &= \frac{(c-1)d}{c(c+d)} - \alpha((c+d)D). \end{aligned}$$

Setting α as $\Theta(1/D)$ results in a function of c and d. For example, picking $c = 2, d = 1, \alpha = 1/(36D)$ results in $p_e = 1/12$, a constant.

Every T-step episode has at least probability p_e of escaping. There are at most SAm_e number of escapes before everything is visited m_e times, so we can bound how many episodes there are until everything is known with high probability. We use Lemma 56 from (Li 2009) to yield $O\left(\frac{1}{p_e}(SAm_e + \ln(1/\delta))\right) = \tilde{O}(SAm_e)$ for the number of episodes. The total timesteps is $\tilde{O}\left(SADm_e\right)$. □

Algorithm 1 PAC-EXPLORE

```
Input: $S, A, R, T, m_e$,
while some $(s, a)$ hasn't been visited at least $m_e$ times do
  Let $s$ be the current state
  if all $a$ have been tried $m_e$ times then
    This is the start of a new $T$-step episode
    Construct $\widehat{M}_K$
    Compute an optimistic $T$-step policy $\widehat{\pi}$ for $\widehat{M}_K$
    Follow $\widehat{\pi}$ for $T$ steps or until reach an unknown s–a
  else
    execute $a$ that has been tried the least
  end if
end while
```

We now bound the sample complexity of ClusterCMBIE.

Theorem 3. *Consider N different MDPs, each with K_N copies, for a total of NK_N MDPs. Given ϵ and δ, ClusterCMBIE will select an ϵ-optimal action for all $K = NK_N$ agents on all but at most*

$$\tilde{O}\Bigg(SAm_eNK_N\left(D - \frac{1}{\epsilon(1-\gamma)^2}\right) + \frac{SAN}{\epsilon(1-\gamma)^2}\left(\frac{S}{\epsilon^2(1-\gamma)^4} + (K_N - 1)\right)\Bigg) \qquad (2)$$

actions, with probability at least $1 - \delta$.

Proof. For phase 1, the idea is to set m_e just large enough so that we can use the confidence intervals to cluster reliably. By our condition, there is at least one state–action pair with a significant gap Γ for any two distinct MDPs, so we just need confidence intervals that are half that size. Using the reward dynamics as an example, an application of Hoeffding's inequality gives $1 - \delta'$ confidence intervals when $m_e = \Omega\left(\frac{1}{\Gamma^2}\log\left(\frac{1}{\delta'}\right)\right)$ of that size. Similar confidence intervals can be derived for transition dynamics. Using a union bound over all state–action pairs, we can detect, with high probability, when they are different, and when they are the same. Another union bound over the $O(N^2)$ checks ensures in high probability that we cluster correctly.

Therefore clustering requires $m_e = \tilde{\Theta}(1/\Gamma^2)$. However recall that PAC-EXPLORE requires $m_e = \tilde{\Omega}\left(SD^2\right)$ and so we set m_e as $\max\left(\tilde{\Theta}(1/\Gamma^2), \tilde{\Theta}(SD^2)\right)$. Then the total sample complexity is the sample complexity of PAC-EXPLORE plus the sample complexity of CMBIE.

$$\tilde{O}\left(SADm_eNK_N\right) + \tilde{O}\left(\frac{SAN}{\epsilon(1-\gamma)^2}\left(\frac{S}{\epsilon^2(1-\gamma)^4} + (K_N - 1) - m_eK_N\right)\right) \tag{3}$$

The $-m_eK_N$ term arises above because the samples that are obtained with PAC-EXPLORE are used immediately in CMBIE to contribute to the total samples needed, so CMBIE needs less samples in the second phase. Combining these terms and rearranging yields the bound. □

We now examine the potential improvement in the sample complexity of clustering CMBIE over standalone agents.

From the CMBIE analysis, we already know if $K_N \leq \frac{S}{\epsilon^2(1-\gamma)^4}$ then we can get a linear speedup for each cluster, resulting in a linear speedup overall if we increase the number of MDPs of each type. The new tradeoff is the overhead in PAC-EXPLORE, specifically in the term $SAm_eNK_N\left(D - \frac{1}{\epsilon(1-\gamma)^2}\right)$, which looks to be a strange tradeoff, but actually what is being compared is the length of an episode. For PAC-EXPLORE, each episode is $\Theta(D)$ steps, whereas for CMBIE, the analysis uses $\Theta(1/(1-\gamma))$ as the episode/horizon length. The additional $1/(\epsilon(1-\gamma))$ comes from the probability of escape for CMBIE, whereas the probability of escape for PAC-EXPLORE is constant. This difference is from the different analyses as CMBIE does not assume anything about the diameter, thus CMBIE is more general than PAC-EXPLORE. We believe analysing CMBIE with a diameter assumption is possible but not obvious. Strictly from the bounds perspective, if D is much smaller than $\frac{1}{\epsilon(1-\gamma)^2}$, then there will not be any overhead. The lack of overhead makes sense as all of the samples in the first exploratory phase are used in the second phase by CMBIE.

However, more significantly, there's an implicit assumption that $K_Nm_e \leq m$ (m is the total number of samples needed for each state–action pair to achieve near-optimality) in the bound. Otherwise if, for example, $m_e = m$, then all the mistakes are made in the PAC-EXPLORE phase with no sharing, so there is no speedup at all. This is where Γ and D matters. Γ must be larger than the $\epsilon(1-\gamma)$ accuracy, as mentioned before, and D must also not be too large, in order to keep $K_Nm_e \leq m$. If $K_Nm_e \leq m$ holds, then after the clustering, all the samples are shared at once. Therefore in many situations we expect the nice outcome that the initial exploration performed for clustering will perform no redundant exploration, and the resulting sample complexity will essentially be the same as CMBIE, where we know in advance which MDPs are identical.

Clustering CMBIE Experiments

We now perform a simple simulation to examine the performance of our ClusterCMBIE algorithm. We assume there are 4 underlying MDPs: the same grid world domain as in our prior experiments, and rotated and reflected versions. This leads to at least one skinny arrow action being distinct between any two different MDPs, and it means very little exploration is required.

The experiments mirrored the CMBIE experiment parameters where applicable, including using the same parameters for CMBIE (which were optimized for single task MBIE), and fixed for all runs. The PAC-EXPLORE algorithm was optimized with $m_e = 1$ and $T = 4$, and fixed for all runs. With these parameters, the first phase of exploration is quick.

Figure 2(a) shows that ClusterCMBIE still achieves a significant increase in how fast it learns the best policy. Figure 2(b) also still shows a significant gain in total rewards as more sharing is introduced. In terms of mistakes, again there is a significant improvement with regards to no clustering/sharding, with only a very small overhead in mistakes as the number of MDPs goes up (Figure 2(c)). We also saw similar results when we used just 2 distinct, underlying MDPs. These results mirror the results for plain CMBIE when it is known all MDPs are identical, owing to how quickly the exploration phase is. All together these results provide preliminary empirical support for our theoretical results, and their performance in practice.

Conclusion and Future Work

We have provided the first formal analysis of RL in concurrent tasks. Our results show that with sharing samples from copies of the same MDP we can achieve a linear speedup in the sample complexity of learning, and our simulation results show that such speedups can also be realized empirically. These also hold under the relaxation that there are a finite number of different types with mild separability and diameter assumptions, where we can quickly explore and cluster identical MDPs together, even without knowing the number of distinct MDPs nor which are identical. This is pleasantly surprising as a linear speedup is the best one may hope to achieve.

Looking forward, one interesting direction is to derive regret bounds for concurrent RL. This may also enable us to enable speedups for even broader classes of concurrent RL: in particular, though in the PAC setting it seems nontrivial to relax the separability assumptions we have employed to enable us to decide whether to cluster two MDPs, pursuing this from a regret standpoint is promising, since clustering similar but not identical MDPs may cause little additional regret. We are also interested in scaling our approach and analysis to efficiently tackle applications represented by large and continuous-state MDPs.

In summary, we have presented solid initial results that demonstrate the benefit of sharing information across parallel RL tasks, a scenario that arises naturally in many important applications.

References

Brafman, R. I., and Tennenholtz, M. 2002. R-max—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3:213–231.

Brunskill, E., and Li, L. 2013. Sample complexity of multi-task reinforcement learning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Evgeniou, T., and Pontil, M. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 109–117.

Jaksch, T.; Ortner, R.; and Auer, P. 2010. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research* 11:1563–1600.

Kakade, S. M. 2003. *On the Sample Complexity of Reinforcement Learning.* . Ph.D. Dissertation, University College London.

Kearns, M. J., and Singh, S. P. 2002. Near-optimal reinforcement learning in polynomial time. *Machine Learning* 49(2–3):209–232.

Lattimore, T.; Hutter, M.; and Sunehag, P. 2013. The sample-complexity of general reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 28–36.

Lazaric, A., and Restelli, M. 2011. Transfer from Multiple MDPs. In *Proceedings of the Neural Information Processing Systems (NIPS)*, 1746–1754.

Lewis, M. 2005. A dynamic pricing approach to customer relationship pricing. *Management Science* 51(6):986–994.

Li, L. 2009. *A Unifying Framework for Computational Reinforcement Learning Theory*. Ph.D. Dissertation, Rutgers University, New Brunswick, NJ.

Silver, D.; Newnham, L.; Barker, D.; Weller, S.; and McFall, J. 2013. Concurrent reinforcement learning from customer interactions. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 924–932.

Strehl, A. L., and Littman, M. L. 2008. An analysis of model-based Interval Estimation for Markov Decision Processes. *Journal of Computer and System Sciences* 74(8):1309–1331.

Strehl, A. L.; Li, L.; and Littman, M. L. 2006. Incremental model-based learners with formal learning-time guarantees. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI-06)*, 485–493.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10(1):1633–1685.

Xue, Y.; Liao, X.; Carin, L.; and Krishnapuram, B. 2007. Multi-task learning for classification with dirichlet process priors. *The Journal of Machine Learning Research* 8:35–63.