Efficient Bayesian Clustering for Reinforcement Learning

Travis Mandel[1], **Yun-En Liu**[2], **Emma Brunskill**[3], **and Zoran Popović**[1,2]
[1]Center for Game Science, Computer Science & Engineering, University of Washington, Seattle, WA
[2]Enlearn™, Seattle, WA
[3]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA
{tmandel, zoran}@cs.washington.edu, yunliu@enlearn.org, ebrun@cs.cmu.edu

Abstract

A fundamental artificial intelligence challenge is how to design agents that intelligently trade off exploration and exploitation while quickly learning about an unknown environment. However, in order to learn quickly, we must somehow generalize experience across states. One promising approach is to use Bayesian methods to simultaneously cluster dynamics and control exploration; unfortunately, these methods tend to require computationally intensive MCMC approximation techniques which lack guarantees. We propose Thompson Clustering for Reinforcement Learning (TCRL), a family of Bayesian clustering algorithms for reinforcement learning that leverage structure in the state space to remain computationally efficient while controlling both exploration and generalization. TCRL-Theoretic achieves near-optimal Bayesian regret bounds while consistently improving over a standard Bayesian exploration approach. TCRL-Relaxed is guaranteed to converge to acting optimally, and empirically outperforms state-of-the-art Bayesian clustering algorithms across a variety of simulated domains, even in cases where no states are similar.

1 Introduction

Developing agents that trade off exploration and exploitation while making decisions under uncertainty is a fundamental problem in AI, with applications in domains such as healthcare, robotics, and education. Typical solutions, based on optimism under uncertainty, treat under-sampled states and actions as high-value to encourage exploration. However, recent breakthroughs have come from observing that a non-optimistic Bayesian posterior sampling approach [Thompson, 1933] can outperform optimistic methods empirically [Chapelle and Li, 2011; Osband *et al.*, 2013], while achieving strong theoretical guarantees [Agrawal and Goyal, 2013; Osband *et al.*, 2013].

Another fundamental problem is that of generalization. We typically know a ground state space, but learning over it directly can be slow due to data sparsity. In many cases, there are similarities between states, which can be exploited to speed learning. Much work seeks to find these similarities in order to aggregate the state space [McCallum, 1996; Chapman and Kaelbling, 1991; Lin and Wright, 2010], and there is a rich literature on other feature-based generalization approaches such as deep neural networks [Mnih *et al.*, 2015].

These problems are usually solved separately. Either we explore/exploit without generalization (e.g. [Osband *et al.*, 2013; Auer and Ortner, 2007]), or we learn to generalize while including some amount of random exploration (e.g. [Timmer and Riedmiller, 2006; McCallum, 1996]). These areas are non-trivial to combine because acting with respect to a representation which is too compact may prevent us from collecting the kind of data necessary to refine it. To do both at once, we need some uncertainty over not just the transitions and rewards given the most likely representation, but also over the representation itself. Ideally, we could tackle exploration and generalization simultaneously in a way that guarantees asymptotically optimal performance, while also ensuring good performance given limited data. However, this is more challenging and has been less well-studied, especially if we desire powerful nonlinear generalization methods such as clustering.

A Bayesian approach to clustering state dynamics might be to use a prior that specifies states which are likely to share parameters, and sample from the resulting posterior to guide exploration. With limited data, this approach will prefer a smaller model and improve initial performance, but performance will continue to improve as more data leads us to consider dissimilar states to be distinct. However, sampling exactly from a clustering prior is intractable, as there are $O(n^n)$ interdependent clusterings to consider among n states. Therefore, past work has used Gibbs sampling, an MCMC technique, to sample approximately from a clustering prior [Asmuth *et al.*, 2009]; unfortunately, these methods are computationally expensive and lack guarantees except in the limit of infinite runtime.

We propose Thompson Clustering for Reinforcement Learning (TCRL), a family of simple-to-understand Bayesian algorithms for reinforcement learning in discrete MDPs with a medium/small state space. TCRL carefully trades off exploration and exploitation using posterior sampling while simultaneously learning a clustering of the dynamics. Unlike MCMC approaches, both variants of TCRL are computationally efficient, run quickly in practice, and require

no parameter tuning. TCRL-Theoretic achieves near-optimal Bayesian regret bounds, while improving empirically over a standard Bayesian exploration approach. And TCRL-Relaxed is guaranteed to converge to optimal behavior while empirically showing substantial improvement over state-of-the-art Bayesian state clustering algorithms across a variety of domains from the literature, even when no states are similar.

2 Setting and Background

We consider Markov decision problems, but depart slightly from the standard Markov Decision Process (MDP) formulation by defining the dynamics in terms of relative outcomes [Leffler *et al.*, 2007; Asmuth *et al.*, 2009]. Relative outcomes are useful in situations where a small number of events can summarize the dynamics, although they can encode any discrete MDP. Specifically, we assume a discrete state space $\mathcal{S}$ and an action set $\mathcal{A}$. In addition, one is given a set of relative outcomes $\mathcal{O}$ such that after taking an action $a \in \mathcal{A}$ from a state $s \in \mathcal{S}$ the agent observes an outcome $o \in \mathcal{O}$. The agent knows the reward function $R(s, a, o)$, which deterministically outputs a scalar value in $[r_{min}, r_{max}]$, and the transition function $T(s, a, o)$, which deterministically outputs a next state s'. However, it does not know the distribution over relative outcomes at each state. Experience comes in episodes of maximum length τ. The goal of the agent is to learn from experience to maximize the sum of rewards over time. For example, in an educational setting the state might be a partial history of problems and responses, an action might be to give a problem, and the outcomes indicate the student response, giving us reward and allowing us to determine how to update the state. In keeping with past posterior sampling work [Osband *et al.*, 2013; Asmuth *et al.*, 2009], we focus on state spaces small enough to plan in exactly.

PSRL Posterior Sampling for Reinforcement Learning (PSRL) [Osband *et al.*, 2013; Strens, 2000] translates posterior sampling [Thompson, 1933], a state-of-the-art exploration/exploitation method [Chapelle and Li, 2011], to reinforcement learning over an MDP. Given a prior distribution over possible MDPs, PSRL samples an MDP from the posterior, solves the sampled MDP, and then runs that policy for an episode. Despite this very general formulation, the existing PSRL work in the non-factored MDP setting [Osband *et al.*, 2013] learns the parameters of each state in $\mathcal{S}$ independently, due to the independent paramater priors used.

3 Related Work

Combining generalization with state-of-the-art exploration A small amount of past work has explored simultaneously generalizing and using state-of-the-art methods to control exploration in a way that provides guarantees. Typically such generalization has been guided by known properties of the environments, such as linear-quadratic systems [Abbasi-Yadkori and Szepesvári, 2011] or factored MDPs with known structure [Osband and Van Roy, 2014]. We are not aware of another approach which combines powerful nonlinear generalization with state-of-the-art exploration in a way that is guaranteed to both run efficiently and converge to optimal behavior.

General State Aggregation Generalization in reinforcement learning is widely studied. Due to space, we discuss the most related work, state aggregation. Typically, these approaches focus on aggregating states instead of state-action pairs, aside from some recent work on aggregating state-action pairs given a fully-specified model [Anand *et al.*, 2015], and work considering homomorphisms among actions when clustering at the state level [Taylor *et al.*, 2009]. State aggregation work often lacks formal guarantees [Lin and Wright, 2010; Timmer and Riedmiller, 2006; Singh *et al.*, 1995]. Limited work has focused on careful exploration while clustering, either in deterministic systems [Timmer and Riedmiller, 2007], when selecting among a small number of models/aggregations [Ortner *et al.*, 2014] or when clustering is used solely to decrease computation [Ortner, 2013].

Further, we focus on clustering dynamics, not states: clustered states share similar relative outcomes, but may have different values and transitions. Thus the focus here is not on speeding planning but on more efficient learning. Related work clustering outcome dynamics either assumes a known clustering [Leffler *et al.*, 2007; Brunskill *et al.*, 2009] or makes additional assumptions, e.g. that states can be clustered in a very small number of ways [Diuk *et al.*, 2009].

Bayesian State Aggregation iPOMDP [Doshi-Velez, 2009] is a Bayesian method to learn in a POMDP environment while growing the state space. iPOMDP lacks guarantees when run for a finite time, is quite computationally expensive, and it is unclear how to leverage a known MDP state space in iPOMDP.

Bayesian RL Work in Bayesian reinforcement learning (e.g. [Guez *et al.*, 2013; Wang *et al.*, 2005]) provides methods to optimally explore while learning an optimal policy. However, these approaches are typically computationally intractable, and are based on maximizing discounted returns across episodes which can lead to incomplete learning [Scott, 2010], in contrast to our approach which is guaranteed to converge to optimal behavior.

BOSS The most related work is Best of Sampled Set (BOSS) [Asmuth *et al.*, 2009] which uses posterior sampling (with added optimism), and an (optional) clustering prior. Its theoretical guarantees are contingent upon drawing exact samples from the posterior, but empirically BOSS only draws approximate samples when using a clustering prior. BOSS clusters dynamics on a state-level, in contrast to our more flexible state-action clustering method. We experimentally compare to an enhanced version of their MCMC-based clustering method.

4 TCRL

Although PSRL (see Section 2) has good theoretical and empirical performance, Osband et al. 2013 uses priors which treat the parameters of each state as independent (unless factored structure is known [Osband and Van Roy, 2014]). This leaves room for improvement by considering richer priors, such as assuming (some) nearby states share similar relative outcomes. For example, honking a horn on a robotic car likely

has the same effect in most places, while the effects of turning are highly state-dependent. A good clustering of dynamics across state-action pairs would help us require less data about honking while preserving the distinction between turning in different locations.

A principled way to approach this problem is to formalize the intuition that many state-action pairs are similar as a clustering prior, and use it in the PSRL framework [Osband *et al.*, 2013]. Ideally we could sample exactly from this clustering prior; however, this is known to be intractable. Past work [Asmuth *et al.*, 2009] used MCMC approaches to draw an approximate sample given such a prior, but this approach is computationally expensive, sensitive to initialization, and lacks guarantees.

Here we present Thompson Clustering for Reinforcement Learning (TCRL), a family of approaches each of which leverage the structure of the state space[1] to efficiently cluster while simultaneously retaining good performance without a need for parameter tuning. The key idea of our TCRL approaches is that they prefer to cluster states that are nearby in the original state space. Although it may be possible to construct examples where this causes TCRL to underperform, we believe this is a good fit for many real-world domains. For example, in e-commerce, a user's preferences are unlikely to drastically change after a single advertisement. Or in education, the state of a student is unlikely to change much after a single problem.

We introduce two specific algorithms which vary in the details of how this structure is used: TCRL-Relaxed and TCRL-Theoretic. However, both of these approaches cluster states separately for each action, ensuring a more flexible representation than an typical state-clustering approach.

4.1 Preliminaries: Choosing among a small number of clusterings

Let us first examine the simplified case of clustering the dynamics of two states, A and B, given some fixed action a. This will form a key building block for our approaches. One can define a prior probability $P(C)$, where C is the event that the two states are clustered (i.e. share identical dynamics given a). Given a set of data D (consisting of (s, a, o) tuples where $s = A$ or $s = B$), we then need to compute $P(D|C)$ and $P(D|\neg C)$. Once these are computed we can compute the posterior probability that these states are clustered[2] :

$$P(C|D) = \frac{P(D|C)P(C)}{P(D|C)P(C) + P(D|\neg C)P(\neg C)}. \quad (1)$$

Associated with the pair (s, a) is a continuous parameter vector $\vec{\theta}$, where the i^{th} component θ_i denotes the probability of generating the i^{th} relative outcome given s and a. Then $P(D|C)$ can be computed as $P(D|C) = \int P(D|\vec{\theta}, C)P(\vec{\theta}|C)d\vec{\theta}$ and similarly for $P(D|\neg C)$. Since $\vec{\theta}$ are the parameters of a categorical distribution over relative outcomes, the conjugate prior is a Dirichlet, and a closed-form solution to this integral for Dirichlet distributions is well-known. For N observations, the Dirichlet has parameters $\alpha_1, \ldots, \alpha_N$, and the integral is $P(D|\alpha_1, \ldots \alpha_N) = \int P(D|\vec{\theta})P(\vec{\theta}|\alpha_1, \ldots \alpha_N)d\vec{\theta}$ which is equal to:

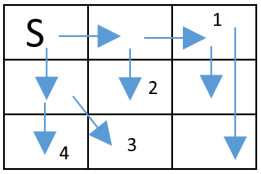
$$\frac{\Gamma(\sum_i \alpha_i)}{\Gamma(\sum_i n_i + \alpha_i)} \prod_{i=1}^{N} \frac{\Gamma(n_i + \alpha_i)}{\Gamma(\alpha_i)}, \quad (2)$$

where n_i is the number of occurrences of the i^{th} outcome in the dataset.

To compute $P(D|C)$, we first sum up the counts of the data over both A and B, that is let $n_i = n_{i,A} + n_{i,B}$, and then compute the probability of the data as in (2). To compute $P(D|\neg C)$, since states A and B are separate, their respective datasets D_A and D_B are conditionally independent given $\neg C$, so we can simply multiply the likelihoods. $P(D_A, D_B|\neg C) = P(D_A|\neg C)P(D_B|\neg C)$, so:

$$P(D_A, D_B|\neg C) = P(D_A|\vec{\alpha_A})P(D_B|\vec{\alpha_B}). \quad (3)$$

So, if we are given two states, we can sample from a simple clustering posterior exactly in constant time. Similarly, if we have a very small number of different clusterings we wish to choose between (e.g. all clustered, predefined half clustered, none clustered), we simply calculate the data likelihood for each cluster using equation (2) and multiply across clusters as in equation (3). However, so far we have not addressed how to scale up in order to perform clustering over a full state space. TCRL-Theoretic and TCRL-Relaxed offer two different solutions to this problem.



Possible Clusterings:
(1),(2),(3),(4) - By Self
(1,2),(3,4) - By Parents
(1,2,3,4) - By Grandparents

Figure 1: The balanced tree TCRL-Theoretic constructs in this gridworld if east children are processed before south children. S is the start state; the arrows go from parents to children in the tree. After constructing the tree, we cluster. Clustering options are shown for the 2nd layer of the tree.

4.2 TCRL-Theoretic

There are $O(n^n)$ interdependent clusterings to consider among n states, so, if we wish to generate an exact sample from a posterior *efficiently*, we must reduce the clustering space. TCRL-Theoretic utilizes the structure of the state space to reduce the space of clusterings in a way that (as we will show experimentally) is sufficient to enable improvements in learning, while retaining near-optimal theoretical guarantees on the expected Bayesian regret. The general approach to reducing the space is to partition the states into

[1] By structure, we refer to properties that can be deduced without any data, specifically the topology of the space as indicated by the transitions and rewards possible due to the relative outcomes, as well as the location of the start state.

[2] Note that this is just an application of Bayes' Rule: We compute $P(D, C)$ and $P(D, \neg C)$ and then normalize. This approach bears a close connection to the Bayes Factor (and likelihood ratio) tests for comparing two competing hypotheses.

independent groups, and then within each group propose a small number of clusterings.

To automatically construct this breakdown a priori, we rely on the structure of the MDP state space. Specifically, our algorithm creates a tree where the nodes are states and the root is the provided start state. The goal is to ensure the tree is fairly balanced, but at the same time ensure that states which are closely connected in the original MDP are closely connected in the tree. To achieve this we traverse the state-space starting from the start state, in a breadth-first fashion, where the traversal follows the transitions that are a priori possible due to the set of possible outcomes. However, since we wish the tree to be roughly balanced and binary, the procedure is complicated by the fact that each state may not have exactly two children. Therefore we use a subroutine getTwoChildren(p) which initiates another breadth-first traversal starting at state p looking for two unseen children to add as children of p. One complication here is that, since transitions may be one-directional, we may orphan nodes. To avoid this we first make sure to include likely orphans as children (immediate children of p without a link back to p). Any remaining orphans are added in a postprocessing step. See Figure 1 for an example, and algorithm 4 in section 8 for pseudocode detailing how the tree is constructed.

Given a tree, a natural way to reduce the number of clusterings is to cluster only within each depth. However, further reduction is needed as each depth is still too large to allow us to consider all possible clusterings within it. Given that the tree structure tends to put nodes which are closer in the original state space graph closer in the tree (for example see Figure 1), it generally seems reasonable to say that states which have the same parent are more likely to cluster than nodes which only share the same grandparent, etc. So for each depth d we consider only $O(\log n)$ clusterings: clustered by depth d (unclustered), clustered by depth $d-1$ (states with same parents clustered together), clustered by depth $d-2$ (states with same grandparents clustered together),... clustered by depth 0 (everything clustered), which allows us to naturally interpolate between different levels of clustering at each depth. See Figure 1 for an example. For each depth d we use a prior on this clustering scheme which puts 0.5 probability on unclustering and $\frac{0.5}{d}$ on clustering by the other depths. The reason for putting more probability mass on the unclustered hypothesis is to reduce the risk of clustering too aggressively.

Next we calculate the probability of each of the clusterings at each depth using Bayes rule, as explained in Section 4.1. After building the tree and sampling a clustering, we sample the parameters of the MDP and solve it to produce the policy to run during the next episode. For details see Algorithm 1.

The runtime of the clustering step is O($\log^2 n$), allowing the procedure to remain highly efficient and scalable. Note that a large part of the benefit is due to the independence among clustering decisions between depths: the total number of possible clusterings is $O((\log n)^{\log n})$, so this does not reduce to making the total number of clusterings very small, unlike some previous work [Diuk *et al.*, 2009; Vien *et al.*, 2013; Ortner *et al.*, 2014].

Note that TCRL-Theoretic samples exactly from the posterior at each step. This allows us to bound the expected *regret*

Algorithm 1 TCRL-Theoretic

```
Input: MDP with unknown dynamics, initial state I
tree = BUILDBALANCEDTREE(I)
for e = 1 to ∞ do
    for a ∈ A do
        C_a = DOCLUSTER(tree)
        Sample parameters θ_ac for each cluster c ∈ C_a
    Create MDP M, using parameters θ_ac
    Solve M to get an optimal policy π
    Run π for an episode and update posterior
procedure DOCLUSTER(tree)
    clustersF = {}
    for depth = 1 to tree.maxDepth do
        probs = {}, clusters = {}
        depthNodes = tree.getStatesAtDepth(depth)
        for d = 0 to depth do
            prior = 0.5/depth
            if d = depth then prior = 0.5
            clusters[d]={}
            for s in depthNodes do
                ancestor = tree.getAncestorAtDepth(s, d)
                clusters[d][ancestor].add(s)
            probs[d] = prior * ∏_{a∈clusters[d].keys()} a.logLikelihood(data)
            (likelihood computed per (2) with ∀i, α_i = 1)
        dF ~ normalize(probs)
        clustersF.add(clusters[dF])
    return clustersF
end procedure
```

of TCRL-Theoretic. We define regret as in Osband et al. [Osband *et al.*, 2013]: $R(T) = \sum_{e=1}^{\lceil T/\tau \rceil} V^* - V_{\pi_e}$, where V^* is the optimal expected per-episode reward, and V_{π_e} is the expected reward of running the policy generated at episode e, π_e, for one episode. Now we present the following theorem.

Theorem 4.1. *The Bayesian regret of TCRL-Theoretic is at most:*

$$O((r_{max} - r_{min})\tau|\mathcal{S}|\sqrt{|\mathcal{A}|T\log(|\mathcal{S}||\mathcal{A}|T)}).$$

Theorem 4.1 does not require a separate proof as it follows directly from the guarantees of PSRL [Osband *et al.*, 2013], the only difference being scaling to $[r_{min}, r_{max}]$ due to the rewards not necessarily being in $[0, 1]$. Note that this is a Bayesian bound, which means that it applies assuming MDPs are truly drawn from the (in our case clustering) prior distribution; however, there is a close connection between frequentist and Bayesian regret [Osband *et al.*, 2013].

4.3 TCRL-Relaxed

In order to sample exactly from the posterior (and thus retain the regret bounds of PSRL) while remaining efficient, TCRL-Theoretic restricts the set of clusterings severely, by only considering clusterings within each depth. A richer set of possible clusterings would likely improve performance if we

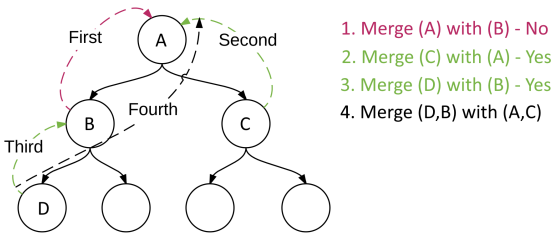


Figure 2: The first 4 clustering decisions that could be made by TCRL-Relaxed on an example DAG. In this run TCRL-Relaxed chose not to cluster first pair of states.

could sample over them efficiently. TCRL-Relaxed is an approximate posterior sampling method that lacks regret guarantees, but it allows more powerful clusterings, remains computationally efficient, and is guaranteed to converge to the optimal policy in the true (unclustered) MDP.

TCRL-Relaxed uses a semi-greedy agglomerative clustering approach. For each action it iterates through pairs of states, clustering them based on the procedure for clustering two states (Section 4.1). Note that, in contrast to TCRL-Theoretic, this is an approximation because the clustering decisions of future states are dependent on how we clustered past states, but we do not integrate over future clustering decisions when calculating the probability of clustering a pair.

In order to define the sequence of pairwise clustering decisions, we exploit structure in the state space to build a directed acyclic graph (DAG). To construct this DAG, we traverse the state space in a breadth-first fashion, ignoring states we have seen before, following edges based on the possible transitions (as defined by relative outcomes) from any given state. All parents of a state s which we reach before s are considered its parents in the DAG (and therefore, their ancestors are also the ancestors of s). Note that unlike TCRL-Theoretic, we do not attempt to balance the DAG. See algorithm 3 in section 8 for full details about how the DAG was constructed.

For each node, we walk through its ancestors in the DAG, testing whether the node (and all others clustered with it so far) can be added to the cluster of the next ancestor. See Figure 2 for an example, and Algorithm 2 for the pseudocode. Although clustering only with ancestors has its limitations (siblings cannot be directly clustered, although they can indirectly cluster through a common parent), ancestors are likely to have much more data than siblings as they are closer to the start state, so the benefit of clustering with them is larger. Further, clustering only with ancestors can often greatly increase computational efficiency (from O(n^2) to O($n \log n$) if the state space is naturally balanced).

Theorem 4.2. *TCRL-Relaxed converges to optimal behavior in the limit of infinite experience.*

Proof. Recall we are in the finite-horizon setting, where experience comes in episodes of maximum length τ. First, we show that if, under an optimal policy π^* in the true MDP, state-action pair (s,a) could be visited, it will be visited an infinite number of times by TCRL-Relaxed. Consider the decision of whether to cluster state A with a cluster C. Consider the case where C is composed of one or more states that share true parameters θ_C, but these are different from A's true parameters θ_A. Observe that when making this decision, the probability of choosing not to cluster is always non-zero and goes to one as more data is collected from A and C due to the consistency of the Bayesian hypothesis test. Since this probability is nonzero and approaches one for every A and C, we must sample a *valid* clustering (that is, one where no states with truly different parameters are clustered) an infinite number of times. As in PSRL, once we have a clustering we sample a policy according to the probability it is optimal given that clustering. Since, given a valid clustering, the probability of π^* being optimal is always non-zero and should converge to some non-zero value (1 if it is unique, $1/q$ if there are q optimal policies), then across an infinite number of rounds, π^* will be sampled an infinite number of times. So any (s,a) occurring in π^* will be sampled an infinite number of times.

Next we show that the distribution over outcomes at *any* (s,a) pair in the sampled MDP will either converge to the true distribution or be sampled a finite number of times. The only way that the outcome distribution of (s,a) does not converge to its true value in the limit of infinite samples is if it is clustered with at least one pair (s',a) with different parameters. However, if (s',a) is also sampled an infinite number of times, the Bayesian hypothesis test will uncluster the two states with probability approaching 1. And if (s',a) is sampled only a finite number of times, the samples will be overwhelmed by the samples from (s,a) and thus still converge to the true values.

At each episode we select one of a finite number of pos-

Algorithm 2 TCRL-Relaxed

```
Input: MDP with unknown dynamics, initial states I
for e = 1 to ∞ do
    for a ∈ A do
        C_a = DOCLUSTER(I)
        Sample parameters θ_ac for each cluster c ∈ C_a
    Create MDP M, using parameters θ_ac
    Solve M to get an optimal policy π
    Run π for an episode and update posterior
procedure DOCLUSTER(I)
    dag = BUILDDAG(I), clusters = {}
    for s ∈ dag.TraverseBreadthFirst() do
        clusters[s] = [s], aClusters = []
        for anc ∈ dag.getAncestors(s) do
            aClusters.union(clusters[anc])
        for ac ∈ aClusters do
            if SAMPLEMERGE(ac, clusters[s]) then
                ac.add(clusters[s])
                for s' in clusters[s] do:
                    clusters[s'] = ac
    return clusters
end procedure
procedure SAMPLEMERGE(s1,s2)
    Compute posterior probability using data based on eqns (2) and (1), using a prior of 0.5 and ∀i, α_i = 1
    Return true with posterior probability, false otherwise
end procedure
```

sible deterministic policies to deploy. So for a contradiction, assume that in the limit of infinite data, with probability $\epsilon > 0$ we choose a policy $\hat{\pi}$ with expected reward worse than π^*. In this case, we know that since $\hat{\pi}$ is sampled an infinite number of times, the outcome distributions for the (s,a) pairs that could be visited under $\hat{\pi}$ will converge to the true distributions. Therefore, the value of $\hat{\pi}$ in the sampled MDP will converge to its true value. However, as we showed above, any (s,a) pairs that appear in π^* will also be sampled an infinite number of times, so the value of π^* will converge to its true value in the sampled MDP. Since the value of π^* is by definition greater than $\hat{\pi}$, and the planning step picks the optimal policy in the sampled MDP, we will converge to putting zero probability on $\hat{\pi}$, a contradiction. □

5 Simulation Results

Here we test performance on several MDP environments, averaging results over 100 runs unless otherwise indicated.

Baselines First, to evaluate the usefulness of clustering, we compare against a straightforward tabular approach which does not generalize between states: PSRL with independent parameter priors. Second, we compare TCRL to a powerful MCMC clustering method similar to that used in BOSS [Asmuth *et al.*, 2009]. Specifically, we take the Chinese restaurant process (CRP) prior used as the BOSS cluster prior [Asmuth *et al.*, 2009], and improve it by clustering states separately for each action, so that this approach, like TCRL, performs clustering of state-action pairs. Given this prior and an action, a typical Gibbs sampling approach is used following Asmuth et al. 2009, where we repeatedly iterate through states, sampling from the conditional distribution over clusters for the current state given that the clustering of the other states is fixed. As with the TCRL approaches, after we sample a clustering we sample the parameters from each state to get an MDP, and then solve the MDP to obtain the policy to use for the next episode.

This baseline MCMC clustering method has several parameters; unfortunately, it is not always clear how to set them. First α, the CRP concentration parameter must be set. We choose the value of 0.5 recommended[3] by Asmuth et al. 2009. Second, since MCMC has no clear termination criterion, we must choose how long to run it. Since fast runtime is often very important, we control for time by, at each iteration, first running TCRL-Relaxed, and then running MCMC for the amount of time TCRL-Relaxed took. Note that controlling for time is always a somewhat inexact science (we are not claiming to have fully optimized either MCMC or TCRL methods). For environments similar to those studied by Asmuth et al. 2009, we also compare running MCMC for the fixed number of iterations (i.e. sweeps across all variables) recommended[4] by Asmuth et al. 2009. Finally, one must set the initialization for MCMC. We compare two logical alternatives: Either all states are initialized in a single cluster (MCMC-Together), or all states are initialized in separate clusters (MCMC-Separate). We label the fixed iterations variant MCMC-Separate-Long.

Method	RiverSwim	MarbleMaze
MCMC-Separate-Long	290.38 ms	1253.47 ms
TCRL-Relaxed	0.47 ms	4.72 ms
TCRL-Theoretic	0.23 ms	1.28 ms

Table 1: Average time to sample a single clustering on RiverSwim and MarbleMaze

Riverswim RiverSwim [Strehl and Littman, 2008] is a 6-state chain MDP previously used to showcase PSRL [Osband *et al.*, 2013]. For a diagram and complete description of the environment, see [Osband *et al.*, 2013]. We used a horizon of 20 and defined 5 relative outcomes for moving left and right or staying with some reward. The results n this environment (Figure 3) show that MCMC-Together performs poorly, being unable to uncluster sufficiently given limited time. Interestingly, despite being given more runtime, MCMC-Separate-Long performs worse than MCMC-Separate, perhaps because the short runtime means MCMC-Separate is more likely to avoid over-clustering. TCRL-Theoretic matches the performance of the best MCMC approach, while TCRL-Relaxed performs uniformly better than any MCMC approach. In Table 1, we found the TCRL approaches to be over 600x faster in this environment than a typical MCMC approach.

MarbleMaze MarbleMaze [Asmuth *et al.*, 2009; Russell *et al.*, 1994] is a 36-state gridworld MDP previously used to showcase the benefit of MCMC clustering in BOSS [Asmuth *et al.*, 2009]. For a diagram and complete description of the environment, see [Asmuth *et al.*, 2009]. We used a horizon of 30 and a set of 5 outcomes denoting whether the agent moved in each cardinal direction or hit a wall (in keeping with past work [Asmuth *et al.*, 2009], the coordinates of the goal and pits are assumed to be known). In Figure 4 we see that TCRL-Relaxed outperforms PSRL and the time-limited MCMC variants by a large margin, while MCMC-Separate-Long appears to slightly outperform it. However, in Table 1 we see in our experiment that variant took over 200x times longer than TCRL-Relaxed. By the end of 1000 episodes, TCRL-Theoretic also outperforms both time-limited MCMC variants. Interestingly, we see that MCMC-Together starts off much better than MCMC-Separate (and PSRL), but over time begins to flatten off, and by 500 episodes MCMC-Separate and PSRL are poised to overtake it. This is likely because aggressive clustering boosts early performance, but in order to reach optimal performance more and more distinctions are needed, which MCMC-Together is unable to make given limited time. On the other hand, MCMC-Separate does not appear to improve much over PSRL, likely because it has insufficient time to identify useful clusterings.

SixArms Six Arms [Strehl and Littman, 2008] is a 7-state MDP arranged in a spoke pattern, where each of the six outer states has very different dynamics/rewards, which presents a challenge for an approach based on clustering. For a diagram and environment description, see [Strehl and Littman, 2008]. We used a horizon of 10 and 14 relative observations: mov-

[3] 0.5 was recommended for Chain (similar to Riverswim). We found 0.5 to work better in MarbleMaze than the recommended value of 10.0, and to avoid tuning α, we fix $\alpha = 0.5$.

[4] We used 500 iterations in Riverswim (recommended for Chain, a similar environment), and 100 as recommended for Marble Maze.

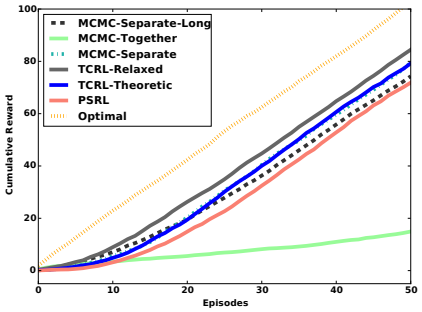


Figure 3: RiverSwim Results.

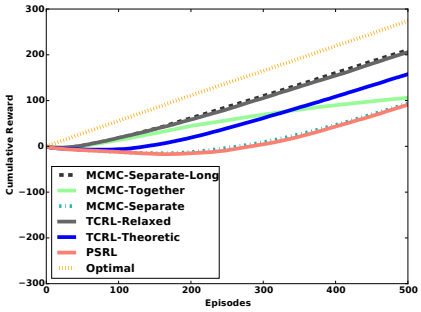


Figure 4: Marble Maze Results.

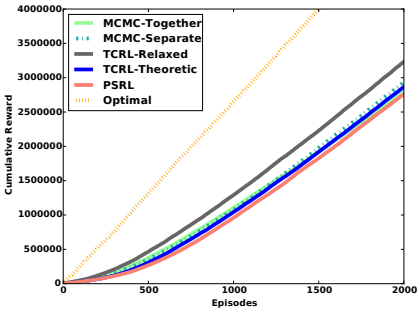


Figure 5: SixArms Results.

ing to each outer state, staying with different rewards, and moving to the inner state. As expected, in Figure 5 we see that TCRL-Theoretic and the MCMC approaches struggle to improve over PSRL. However, TCRL-Relaxed shows a major improvement over PSRL and both time-limited MCMC variants. The fact that a clustering approach is able to improve over PSRL in the setting illustrates the potential benefit of state-action clustering: even though no states are similar in this domain, many (s,a) pairs are. We also observed that TCRL-Relaxed frequently clusters (s,a) pairs that have different dynamics if they are in low-value regions of the state space where we have less data and distinguishing dynamics is less important, allowing it to focus on improving performance in the higher-value areas.

200-state environment Next we examined a larger 200-state gridworld featuring one-dimensional walls [Johns and Mahadevan, 2007], with an added start state in the bottom-right corner. For a diagram see [Johns and Mahadevan, 2007]. The setup is similar to MarbleMaze, except that we chose the reward to be 100 for reaching the goal and -1 for each step taken, and since the problem was harder we used a longer horizon of 50 and averaged over 20 (instead of 100) runs. In Figure 6, the approaches which can aggressively cluster (TCRL-Relaxed, MCMC-Together, and TCRL-Theoretic) greatly outperform those that do little or no clustering (PSRL and MCMC-Separate), demonstrating the importance of generalization in larger environments. Here MCMC-Together performs similarly to TCRL-Relaxed, probably because most states share identical dynamics in this mostly empty gridworld and thus a highly nuanced clustering is not required.

6 Discussion

Overall, TCRL-Relaxed leverages the structure of the state space along with its pairwise Bayesian clustering to outperform PSRL and either choice of time-limited MCMC initialization, with the exception of the 200-state grid where it matched the best MCMC variant. TCRL-Theoretic, although worse in performance than TCRL-Relaxed, always improves over PSRL, typically by a substantial amount, and has stronger guarantees. In contrast to TCRL which requires no parameter tuning, MCMC is extremely sensitive to initialization when time-limited. In the 200-state grid, initializing together does well and initializing separately does very poorly, but the opposite is true in RiverSwim. Using a large

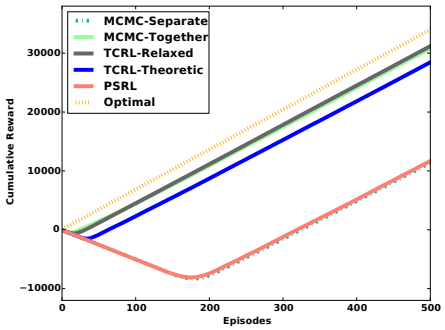


Figure 6: Results in the 200-state environment.

number of MCMC iterations (as specified by [Asmuth *et al.*, 2009]) can improve performance, but in our experiments doing this takes 200x-600x longer than TCRL-Relaxed. Further, our MarbleMaze results show variants of MCMC can level off before reaching optimal performance, in contrast to TCRL. Finally, we find it encouraging that in SixArms, where we expected clustering to have little benefit, TCRL-Relaxed is able to improve.

7 Conclusion

In this work we developed TCRL-Theoretic and TCRL-Relaxed, two Bayesian approaches which allow us to generalize across states via clustering the dynamics while carefully controlling exploration. Unlike typical MCMC approaches, our techniques have no parameters to tune, and exploit the structure of the state space to remain computationally efficient. Further, TCRL-Relaxed is guaranteed to achieve optimal performance, while TCRL-Theoretic has stronger guarantees of achieving near-optimal Bayesian regret. We showed in simulation that our approaches improve over PSRL, with TCRL-Relaxed typically also outperforming both time-limited MCMC approaches by a substantial margin. One limitation we hope to address in future work is that TCRL struggles to scale to very large state spaces, in part due to the complexity of the required offline planning step. Approximate or online planning may be helpful here. Also, it would be interesting to investigate larger or continuous outcome spaces.

8 Appendix

Algorithm 3 Subroutine for TCRL-Relaxed

```
procedure BUILDDAG(I)
    fringe = [I], seen = [], ancestors = {}, clusters = {}
    while fringe is not empty do
        s = fringe.pop()
        if s ∈ seen then continue
        seen.add(s);
        for child ∈ s.children() do        ▷ s.children() returns all children reachable after taking some action and observation.
            if child ∈ seen then continue
            ancChild = ancestors[s].add(s)
            if child ∈ fringe then
                ancestors[child].add(ancChild)
            else
                fringe.add(child)
                ancestors[child] = ancChild
end procedure
```

Algorithm 4 Subroutine for TCRL-Theoretic

```
procedure BUILDTREE(I)
    tree.setRoot(I), fringe = [I], seen = [], backptrs = []
    while fringe is not empty do
        s = fringe.removeFirst()
        if seen.contains(s) then continue else seen.add(s)
        children = GETATLEASTTWOCHILDREN(s, seen, fringe)
        tree.setChildren(s, children), fringe.add(children)
    while some state not in tree do
        for state in tree do
            if state.children() not all in tree then
                tree.UnionChildren(state, state.children())
    return tree
end procedure
procedure GETTWOCHILDREN(parent, seenOuter, fringeOuter)
    Summary: Traverses the graph to try to get exactly two new children, may go over to handle likely orphans (immediate
children without a link back to parent)
    children = parent.children(), ret = {}, fringe = {children}, seen = {}, workingOnSubChildren = false
    while fringe.size() > 0 and ret.size() < 2 do
        newFringe = {}
        for s in fringe do
            if seen.contains(node) or (!workingOnSubChildren
                and re.size() ≥ 2 and
                node.children().contains(parent)) then
                continue
            if !seenOuter.contains(s) and
               !fringeOuter.contains(s) then
                ret.add(s)
                if ret.size() ≥ minDegree and workingonSubChildren then
                    break
            seen.Add(s)
            for subChild in s.getChildren() do
                newFringe.Add(subChild)
        workingOnSubChildren = true, fringe = newFringe
end procedure
```

Acknowledgements

This work was supported by the NSF BIGDATA grant No. DGE-1546510, the Office of Naval Research (ONR) Young Investigator Award N00014-16-1-2241, ONR grant N00014-12-C-0158, the Bill and Melinda Gates Foundation grant OPP1031488, the Hewlett Foundation grant 2012-8161, Adobe, Google, and Microsoft.

References

[Abbasi-Yadkori and Szepesvári, 2011] Yasin Abbasi-Yadkori and Csaba Szepesvári. Regret bounds for the adaptive control of linear quadratic systems. In *COLT*, pages 1–26, 2011.

[Agrawal and Goyal, 2013] Shipra Agrawal and Navin Goyal. Further optimal regret bounds for Thompson sampling. In *AISTATS*, pages 99–107, 2013.

[Anand *et al.*, 2015] Ankit Anand, Aditya Grover, Mausam Mausam, and Parag Singla. ASAP-UCT: abstraction of state-action pairs in UCT. In *IJCAI*, pages 1509–1515. AAAI Press, 2015.

[Asmuth *et al.*, 2009] John Asmuth, Lihong Li, Michael L Littman, Ali Nouri, and David Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *UAI*, pages 19–26. AUAI Press, 2009.

[Auer and Ortner, 2007] P Auer and R Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. *NIPS*, 19:49, 2007.

[Brunskill *et al.*, 2009] Emma Brunskill, Bethany R Leffler, Lihong Li, Michael L Littman, and Nicholas Roy. Provably efficient learning with typed parametric models. *JMLR*, 10:1955–1988, 2009.

[Chapelle and Li, 2011] Olivier Chapelle and Lihong Li. An empirical evaluation of Thompson sampling. In *NIPS*, pages 2249–2257, 2011.

[Chapman and Kaelbling, 1991] David Chapman and Leslie Pack Kaelbling. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *IJCAI*, volume 91, pages 726–731, 1991.

[Diuk *et al.*, 2009] Carlos Diuk, Lihong Li, and Bethany R Leffler. The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning. In *ICML*, pages 249–256. ACM, 2009.

[Doshi-Velez, 2009] Finale Doshi-Velez. The infinite partially observable Markov decision process. In *NIPS*, pages 477–485, 2009.

[Guez *et al.*, 2013] Arthur Guez, David Silver, and Peter Dayan. Scalable and efficient bayes-adaptive reinforcement learning based on Monte-Carlo tree search. *JAIR*, pages 841–883, 2013.

[Johns and Mahadevan, 2007] Jeff Johns and Sridhar Mahadevan. Constructing basis functions from directed graphs for value function approximation. In *ICML*, pages 385–392. ACM, 2007.

[Leffler *et al.*, 2007] Bethany R Leffler, Michael L Littman, and Timothy Edmunds. Efficient reinforcement learning with relocatable action models. In *AAAI*, volume 7, pages 572–577, 2007.

[Lin and Wright, 2010] Stephen Lin and Robert Wright. Evolutionary tile coding: An automated state abstraction algorithm for reinforcement learning. In *SARA*, 2010.

[McCallum, 1996] Andrew Kachites McCallum. Learning to use selective attention and short-term memory in sequential tasks. In *SAB*, volume 4, page 315. MIT Press, 1996.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Ortner *et al.*, 2014] Ronald Ortner, Odalric-Ambrym Maillard, and Daniil Ryabko. Selecting near-optimal approximate state representations in reinforcement learning. In *Algorithmic Learning Theory*, pages 140–154. Springer, 2014.

[Ortner, 2013] Ronald Ortner. Adaptive aggregation for reinforcement learning in average reward Markov decision processes. *Annals of Operations Research*, 208(1):321–336, 2013.

[Osband and Van Roy, 2014] Ian Osband and Benjamin Van Roy. Near-optimal reinforcement learning in factored MDPs. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *NIPS*, pages 604–612. Curran Associates, Inc., 2014.

[Osband *et al.*, 2013] Ian Osband, Dan Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *NIPS*, pages 3003–3011, 2013.

[Russell *et al.*, 1994] Stuart Russell, Peter Norvig, et al. *Artificial Intelligence A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ, 1994.

[Scott, 2010] Steven L Scott. A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

[Singh *et al.*, 1995] Satinder P Singh, Tommi Jaakkola, and Michael I Jordan. Reinforcement learning with soft state aggregation. *NIPS*, pages 361–368, 1995.

[Strehl and Littman, 2008] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[Strens, 2000] Malcolm Strens. A Bayesian framework for reinforcement learning. In *ICML*, pages 943–950, 2000.

[Taylor *et al.*, 2009] Jonathan Taylor, Doina Precup, and Prakash Panagaden. Bounding performance loss in approximate MDP homomorphisms. In *NIPS*, pages 1649–1656, 2009.

[Thompson, 1933] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

[Timmer and Riedmiller, 2006] Stephan Timmer and M Riedmiller. Abstract state spaces with history. In *NAFIPS*, pages 661–666. IEEE, 2006.

[Timmer and Riedmiller, 2007] Stephan Timmer and Martin Riedmiller. Safe Q-learning on complete history spaces. In *Machine Learning: ECML 2007*, pages 394–405. Springer, 2007.

[Vien *et al.*, 2013] Ngo Anh Vien, Wolfgang Ertel, Viet-Hung Dang, and TaeChoong Chung. Monte-Carlo tree search for Bayesian reinforcement learning. *Applied intelligence*, 39(2):345–353, 2013.

[Wang *et al.*, 2005] Tao Wang, Daniel Lizotte, Michael Bowling, and Dale Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *ICML*, pages 956–963. ACM, 2005.