Representation discovery for MDPs using bisimulation metrics

Sherry Shanshan Ruan and **Gheorghe Comanici** and **Prakash Panangaden** and **Doina Precup**
School of Computer Science,
McGill University, Montreal, QC, Canada
{sherry, gcoman, prakash, dprecup}@cs.mcgill.ca

Abstract

We provide a novel, flexible, iterative refinement algorithm to automatically construct an approximate state-space representation for Markov Decision Processes (MDPs). Our approach leverages bisimulation metrics, which have been used in prior work to generate features to represent the state space of MDPs. We address a drawback of this approach, which is the expensive computation of the bisimulation metrics. We propose an algorithm to generate an iteratively improving sequence of state space partitions. Partial metric computations guide the representation search and provide much lower space and computational complexity, while maintaining strong convergence properties. We provide theoretical results guaranteeing convergence as well as experimental illustrations of the accuracy and savings (in time and memory usage) of the new algorithm, compared to traditional bisimulation metric computation.

Introduction

Solving large sequential decision problems modeled as Markov Decision Processes (MDPs) requires the use of approximations to represent the state space. Popular approximation methods include state aggregation, linear function approximation and kernel-based methods. In this paper we are mainly interested in state aggregation, in which the state space is partitioned into disjoint subsets and values are associated with each partition. The goal is to construct a partition incrementally, in such a way as to provide a good approximation to the true value function. One approach for this problem is to use bisimulation relations (Givan, Dean, and Greig 2003), also known as MDP homomorphisms (Ravindran and Barto 2002), or their relaxation as *bisimulation metrics* (Ferns, Panangaden, and Precup 2004). Bisimulation metrics in particular are attractive because they allow quantifying the approximation error for *any* state space partitioning, or more generally, any linear function approximator (Comanici and Precup 2012). However, bisimulation metric computation is very expensive (in the worst case, more expensive than performing dynamic programming in the original state space). Indeed, recent work (Ferns and Precup 2014) has shown that computing the metric amounts to solving an MDP resulting from a coupling of the state space with itself; such a coupling has size quadratic in the number of states.

In this paper, we tackle this problem by proposing a significant improvement in how bisimulation metrics are computed. While we present our results in the context of MDPs, the algorithmic ideas can also benefit the verification community, which relies on bisimulation metrics in order to automatically verify the concordance of a model with a specification.

Our approach constructs an iteratively improving sequence of state space partitions, which converges in the limit to the bisimulation relation, just like previous algorithms. We prove that at each step the error of the value function computed over this partition (compared to the true optimal value function) is bounded. Since at each step, the value function approximation is computed over partitions rather than states, this approach can generate substantial space and computation time savings, as illustrated in our experiments described later.

The second contribution of the paper consists in an algorithm for asynchronous updates of the metric and the representation. We provide theoretical conditions which allow computational effort to be focused on parts of the state space where changes are happening rapidly, similar to successful asynchronous or distributed dynamic programming techniques such as Bertsekas and Tsitsiklis (1996), Bertsekas and Castanon (1989), Moore and Atkeson (1993) . Empirical results illustrate the use of heuristics that can substantially speed up the computation.

Markov Decision Processes

An MDP is a tuple (S, A, P, R) where S is a finite state space S, A is a finite action space, P is a transition probability function $P : S \times A \times S \to [0, 1]$, with $P^a_{ss'}$ giving the probability that the system will end up in state s' when the action a is performed in state s, and $R : S \times A \to \mathbb{R}$ is the immediate reward function, with R^a_s giving the immediate reward for performing action a in state s.

The main objective of MDP solvers is to compute value functions for different *policies*, *i.e.,* strategies for choosing actions. A policy is denoted by $\pi : S \times A \to [0, 1]$ where π^a_s is the probability of choosing action a in state s. We will define below *the value function of* π, which is dependent on a

discount factor γ ($0 \leq \gamma < 1$), $R^\pi_s = \sum_a \pi^a_s R^a_s$, and $P^\pi_{ss'} = \sum_a \pi^a_s P^a_{ss'}$. The value function is given by the following expectation over sample trajectories $X_1, X_2, X_3, ..., X_n, ...$:

$$V^\pi(s) = E\left[\sum_{i=1}^{\infty} \gamma^{i-1} R^\pi_{X_i} | X_1 = s\right] = R^\pi_s + \gamma \sum_{s'} P^\pi_{ss'} V^\pi(s')$$

Most algorithms for solving MDPs either use the model (R, P) to find V^π if it is available and allowed by the size of the problem, or estimate V^π using samples (s, a, r, s'). In both cases, the computation of V^π is performed in the space $\mathcal{F}_S$ of real valued functions over S, i.e., $\mathcal{F}_S := \{f : S \to \mathbb{R}\}$.

The Bellman equation is a well-known result characterizing the value function $V^\pi \in \mathcal{F}_S$ as the fixed point of the following map $T^\pi : \mathcal{F}_S \to \mathcal{F}_S$ given by

$$\begin{aligned} &T^\pi(f) = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi(f) \\ &\text{with } \mathcal{R}^\pi \in \mathcal{F}_S, \;\; \mathcal{R}^\pi(s) = \textstyle\sum_a \pi^a_s R^a_s \\ &\mathcal{P}^\pi : \mathcal{F}_S \to \mathcal{F}_S, \mathcal{P}^\pi(f)(s) = \textstyle\sum_{a,s'} \pi^a_s P^a_{ss'} f(s') \end{aligned} \quad (1)$$

The value V^* associated with the best policy is the fixed point of the nonlinear Bellman optimality operator T^*:

$$T^*(f) = \max_a \left(\mathcal{R}^a + \gamma \mathcal{P}^a(f)\right) \quad (2)$$

where $\mathcal{R}^a$ and $\mathcal{P}^a$ are associated with the policy choosing a deterministically.

Linear value functions

In most cases, working with the original representation is unfeasible due to the large size of the state space. It is common to work instead with restrictions over the space of functions, $\Phi \subset \mathcal{F}_S$, for which searching for V^π (or its approximation) becomes computationally feasible. It is also quite common to work with linear subspace $\Phi = span(\phi_1, \phi_2, ..., \phi_k)$, for some set of linearly independent finite subset Φ of $\mathcal{F}$ of dimension $k \ll S$.

Accurate representations: If one wants V^π to be an element of Φ, then it is sufficient to guarantee the accuracy of the reward and transition models: $\mathcal{R}^\pi \in \Phi$ and $\forall f \in \Phi, \; \mathcal{P}^\pi(f) \in \Phi$. Several methods have been developed to obtain accurate representations for a given policy π(Parr et al. 2008b; 2008a). The methods described later in this paper are designed to provide accuracy for the model of *any* given policy.

Approximate representations: Given a Φ, *linear fixed point methods* such as TD, LSTD, LSPE (Sutton 1988; Bradtke and Barto 1996; Yu and Bertsekas 2006) can be used to find the *least squares fixed point approximation* V^π_Φ of V^π, which is the fixed point of the alternate operator T^π_Φ:

$$T^\pi_\Phi f := \Pi_\Phi(\mathcal{R}^\pi + \gamma \mathcal{P}^\pi f)$$

where Π_Φ is the orthogonal projection operator on Φ. Using the fact that Π_Φ is itself linear, it is not hard to show that $T^\Phi_\pi f = \Pi_\Phi \mathcal{R}^\pi + \gamma \Pi_\Phi \mathcal{P}^\pi f$; therefore, V^π_Φ is the fixed point of the Bellman operator over the *linear model* $(\mathcal{R}^\pi_\Phi, \mathcal{P}^\pi_\Phi) := (\Pi_\Phi \mathcal{R}^\pi, \Pi_\Phi \mathcal{P}^\pi)$. For computational details using linear models for evaluation, see Parr et al. (2007; 2008) .

Bisimulation relations and metrics

Probabilistic bisimulation is an equivalence relation between states of a process due to Larsen and Skou (1991). It was extended to MDPs with rewards by Givan et al. (2003); the metric analogue is due to Desharnais et al. (1999, 2004) and the extension of the metric to include rewards is from Ferns et al. (2004). Suppose we are given an equivalence relation $\sim$ on the state space S. We say that $\sim$ is a **strong probabilistic bisimulation relation** if the following two conditions are satisfied for any equivalent pair $s \sim s'$ and any choice of action $a \in A$: 1. $R^a_s = R^a_{s'}$; 2. $P^a_s = P^a_{s'}$, as probability measures over equivalence classes of $\sim$. The apparent circularity in this definition can be resolved by a fixed-point argument (Larsen and Skou 1991). The crucial point is that the immediate behaviour of bisimilar states is the same and it stays the same indefinitely.

Note that any equivalence relation $\sim$ determines a representation $B_\sim = \{\phi_i\}_{i=1}^m$, where m is the number of equivalence classes of $\sim$ and each ϕ_i is the characteristic function over equivalence class i. For notational convenience, let $B_\sim \in \{0,1\}^{m \times |S|}$ be the matrix whose columns are the ϕ_i's. It is not hard to check that $B^T e = e$ (where $e(s) = 1, \forall s$). Let δ_s be the characteristic function that is equal to 1 at state s and 0 for all other states. Given any matrix M, $M\delta_s$ and $\delta_s^T M$ are the column and row respectively corresponding to s. Note that $s \sim s'$ iff $B\delta_s = B\delta_{s'}$ (i.e., s and s' have the same representation). Another way to characterize a strong probabilistic bisimulation relation $\sim$ is through the following conditions:

$$\begin{cases} \mathcal{R}^a \in colspan(B_\sim) & \forall a \\ \mathcal{P}^a \phi \in colspan(B_\sim) & \forall a, \forall \phi \in B_\sim \end{cases}$$

Note that checking whether $f \in colspan(B_\sim)$ is equivalent to checking that $f(s) = f(s')$ for all pairs $s \sim s'$, as $B_\sim$ consists of characteristic functions of equivalence classes.

Bisimulation metrics provide relaxations of bisimulation relations, which assign non-negative values to all pairs of states. Two states are bisimilar iff their distance is zero. For other pairs of states that are not bisimilar, the metric quantifies how different the states are from each other. Bounds relating bisimulation metrics to value functions have been derived in Ferns et al. (2004). The metric presented in their work is based on the Monge-Kantorovich metric $\mathcal{T}_d(\mu, \nu)$ for comparing two probability distributions μ and ν using a ground distance d:

$$\begin{aligned} \Lambda(\mu, \nu) &= \left\{\lambda \geq 0 \text{ s.t. } \lambda e = \mu \text{ and } \lambda^T e = \nu\right\} \\ \mathcal{T}_d(\mu, \nu) &:= \max_{\lambda \in \Lambda(\mu,\nu)} \textstyle\sum_{s,s'} (\delta_s^T \lambda \delta_{s'})(\delta_s d \delta_{s'}) \end{aligned}$$

Note that Ferns et al. (2004) does not use couplings; this more recent view is expressed in Ferns and Precup (2014). However, the definition using couplings is much more convenient for proving the results in the following sections. Note that $\mathcal{T}_d(\mu, \nu)$ is 0 if and only if μ and ν match as measures over classes of states at distance d equal to 0. One can use similar measures to compare the reward model, and we use the L_1 measure in the definition below (but any norm on

$\mathbb{R}$ could equally be used):

$$F(d)(s,s') := \max_a \left(|\mathcal{R}^a(s) - \mathcal{R}^a(s')| + \gamma \mathcal{T}_d(\delta_s^T \mathcal{P}^a, \delta_{s'}^T \mathcal{P}^a) \right)$$
$$d^* := \sup_n F^n(0)$$

Two theoretical results about F allow one to compute d^* iteratively: first, the collection of pseudo-metrics is equipped with an order, making it *a complete lattice*[1]; secondly, F is shown to be *a monotone map*. Therefore, $F^n(0)$ increases to the solution d^*, and d^* is a fixed point of F. Moreover, it is not hard to see that if we relate states s and s' iff $d^*(s,s') = 0$, then the corresponding relation is a bisimulation. Additionally, Ferns et al. (2004) state bounds which allow one to assess the quality of representations determining partitions over the state space[2]. Given such a representation B, if one computes the value function V_B^* as the fixed point of $T_B^{\pi^*}$ instead of the fixed point of T^*, the approximation error is bounded as follows:

$$|V_B^*(s) - V^*(s)| \leq \frac{\delta_s^T \Pi_B d^* \delta_s}{1-\gamma} + \frac{\gamma \max_{s'} \delta_{s'}^T \Pi_B d^* \delta_{s'}}{(1-\gamma)^2} \quad (3)$$

Note that for representations that are based on characteristic functions, projection amounts to averaging. That is,

$$\delta_s^T \Pi_B f = \textstyle\sum_i \phi_i(s)(\phi^T f)/(\phi^T \phi)$$

Hence, the value function approximation bounds depend on $\delta_s^T \Pi_B d^* \delta_s$, *the average d^* distance from a state s to all other states that have the same representation* . It should be mentioned that these bounds are presented in terms of state aggregation maps in Ferns et al. (2004). See Parr et al. (2008) for a more detailed discussion on why solving a linear fixed-point solution (i.e., finding the fixed point of T_B^*) amounts to the same solution as solving a linear-model solution of an aggregate model (i.e., finding the fixed point of T^* over the aggregate model). Note that these bounds are minimized by aggregating states which are "close" in terms of the bisimulation distance d^*, i.e., states that are close to being bisimilar. It should also be noted that $V_{B_\sim}^*$ is exactly V^* when $B_\sim$ represents a bisimulation relation. This is because d^* is a pseudo-metric for which distance between states with the same representation is 0, and as a consequence $\delta_s^T \Pi_{B_\sim} d^* \delta_s = 0$.

Similar bounds for normalized features can be found in Comanici and Precup (2011). Moreover, their work presents ways of generating feature-based representations using spectral analysis methods with bisimulation metrics as ground metrics. We now proceed to analyze alternative characterizations of d^* and describe a new algorithmic framework for computing d^*.

[1] A complete lattice is a partially ordered set for which every subset has a greatest lower bound and a least upper bound. This is a crucial property in proving convergence to a fixed point of a monotone operator over such a space.

[2] A representation $\phi(s) \in \{0,1\}$ determines a partition if $\sum_i \phi_i(s) = 1, \ \forall s$. The classes of the partition are determined by the sets $\{\{s \mid \phi(s) = 1\} \mid \phi \in B\}$

Iterative refinement algorithm

In this section we will characterize d^* through a series of metrics over compact representations of the original MDPs. To achieve this, we will first define partitions and show how one can use these compact representations to compute iteratively improving approximations to d^*. A **partition** $\mathcal{B}$ is a basis $\{\phi_i\}_{i=1}^m$ such that $\phi_i \in \{0,1\}$ and $\sum_i \phi_i(s) = 1, \forall s \in S$. We will also use B to denote the matrix whose columns are the elements in $\mathcal{B}$.

Let $\widehat{d^*} = \sup_n B_n^T d_n B_n$, where d_n are metrics over a sequence of partitions B_n, defined inductively as follows:

- $B_0 = \{e\}$ $\ (e(s) = 1$ for every $s)$
- B_n is a partition such that
$$\begin{cases} \mathcal{R}^a \in colspan(B_n) & \forall a \\ \mathcal{P}^a \phi, \phi \in colspan(B_n) & \forall a, \forall \phi \in B_{n-1} \end{cases}$$
- Define the metric d_n based on the Monge-Kantorovich metric over partition B_{n-1} with ground metric d_{n-1}. For $\phi, \phi' \in B_n$, if $\phi(s) = 1$ and $\phi'(s') = 1$, then
$$d_n(\phi,\phi') = \max_a (|\mathcal{R}^a(s) - \mathcal{R}^a(s')| + \gamma \mathcal{T}_{d_{n-1}}(\delta_s^T \mathcal{P}^a B^T, \delta_{s'}^T \mathcal{P}^a B^T))$$
Note that B_n is purposely defined so that the value $d_n(\phi,\phi')$ is the same for any choice of states s and s' with representations ϕ and, respectively ϕ'.

Theorem 1. *Given the sequence $\{B_n, d_n\}_{n=1}^\infty$ defined above, it follows that $B_n^T d_n B_n = F^n(0)$. Then*
$$d^* = \sup_n B_n^T d_n B_n.$$

*This result is **key in designing the new algorithm**.*

Proof. First, we will prove that for a partition B,
$$\mathcal{T}_d(B\mu, B\nu) = \mathcal{T}_{B^T dB}(\mu,\nu)$$
As mentioned before, the Monge-Kantorovich metric is an optimization function over the set of "couplings" of two measures. Remember that the set of couplings $\Lambda(\mu,\nu)$ is the set of maps λ such that $\lambda e = \mu$ and $\lambda^T e = \nu$. Before we continue, it is helpful to point out a few properties of B which will make the derivation much simpler: $B^T e = e$; if $\phi(s) = 1$, $B\delta_s = \delta_\phi$. Now, given $\lambda \in \Lambda(\mu,\nu)$,
$$B\lambda B^T e = B\lambda e = B\mu$$
$$(B\lambda B^T)^T e = B\lambda^T e = B\nu$$
Therefore $B\lambda B^T \in \Lambda(B\mu, B\nu)$. Now,
$$\begin{aligned} &\textstyle\sum_{s,s'} \delta_s^T \lambda \delta_{s'} \delta_s^T (B^T dB) \delta_{s'} \\ &= \textstyle\sum_{\phi,\phi'} \sum_{s,s'} \phi(s)\phi'(s') \delta_s^T \lambda \delta_{s'} (B\delta_s)^T d (B\delta_{s'}) \\ &= \textstyle\sum_{\phi,\phi'} \delta_\phi^T d \delta_{\phi'} \sum_{s,s'} \phi(s) \delta_s^T \lambda \delta_{s'} \phi'(s') \\ &= \textstyle\sum_{\phi,\phi'} (\delta_\phi^T d \delta_{\phi'})(\delta_\phi^T B \lambda B^T \delta_{\phi'}) \end{aligned}$$

$$\begin{aligned} \mathcal{T}_{B^T dB}(\mu,\nu) &= \inf_{\lambda \in \Lambda(\mu,\nu)} \textstyle\sum_{s,s'} \delta_s^T \lambda \delta_{s'} \delta_s^T (B^T dB) \delta_{s'} \\ &= \inf_{\lambda \in \Lambda(\mu,\nu)} \textstyle\sum_{\phi,\phi'} (\delta_\phi^T d \delta_{\phi'})(\delta_\phi^T B \lambda B^T \delta_{\phi'}) \\ &\geq \inf_{\bar{\lambda} \in \Lambda(B\mu, B\nu)} \textstyle\sum_{\phi,\phi'} (\delta_\phi^T d \delta_{\phi'})(\delta_\phi^T \bar{\lambda} \delta_{\phi'}) = \mathcal{T}_d(B\mu, B\nu) \end{aligned}$$

Algorithm 1 partition_declust

Given a partition B and one of its elements $\phi \in B$, we want to find B_ϕ. We initialize it as $B_\phi \leftarrow \emptyset$.
for all s with $\phi(s) = 1$ **do**
  for all $\phi' \in B_\phi$ **do**
    choose s' with $\phi'(s') = 1$
    if $\forall a, \forall \phi'' \in B, (\mathcal{P}^a \phi'')(s) = (\mathcal{P}^a \phi'')(s')$
    then $\phi'(s) \leftarrow 1$ {An equivalent condition can be used when declustering based on reward}
  end for
  if $\phi'(s) = 0,\ \forall \phi' \in B_\phi$
  then add a new element $\widehat{\phi}$ to B_ϕ and set $\widehat{\phi}(s) = 1$
end for

So $\mathcal{T}_d(B\mu.B\nu) \leq \mathcal{T}_{B^T dB}(\mu, \nu)$. Now we will prove the opposite. Define $\widehat{B}_\mu(s, \phi) := \phi(s)\mu(s)/(B\mu)(\phi)$. Note that $B\widehat{B}_\mu = I$ and $\widehat{B}_\mu B\mu = e$. Now let $\lambda \in \Lambda(B\mu, B\nu)$,

$$\widehat{B}_\mu \lambda \widehat{B}_\nu^T e = \widehat{B}_\mu \lambda \widehat{B}_\nu^T B^T e = \widehat{B}_\mu \lambda e = \widehat{B}_\mu B\mu = \mu$$
$$(\widehat{B}_\mu \lambda \widehat{B}_\nu^T)^T e = \widehat{B}_\nu \lambda^T \widehat{B}_\mu^T e = \nu$$

Therefore $\widehat{B}_\mu \lambda \widehat{B}_\nu^T \in \Lambda(\mu, \nu)$. Now,

$$\begin{aligned}
&\textstyle\sum_{\phi,\phi'} \delta_\phi^T \lambda \delta_{\phi'} \delta_\phi^T d \delta_{\phi'} \\
=&\textstyle\sum_{\phi,\phi'} (\delta_\phi^T \lambda \delta_{\phi'} \delta_\phi^T d \delta_{\phi'})(\delta_\phi^T B \widehat{B}_\mu \delta_\phi)(\delta_{\phi'}^T \widehat{B}_\nu^T B^T \delta_{\phi'}) \\
=&\sum_{\phi,\phi'} (\delta_\phi^T \lambda \delta_{\phi'} \delta_\phi^T d \delta_{\phi'}) \sum_{s,s'} \phi(s)\phi(s')(\delta_s^T \widehat{B}_\mu \delta_\phi)(\delta_{\phi'}^T \widehat{B}_\nu^T \delta_{s'}) \\
=&\sum_{\phi,\phi'} \delta_\phi^T d \delta_{\phi'} \sum_{s,s'} \phi(s)\phi(s')(\delta_s^T \widehat{B}_\mu (\delta_\phi \delta_\phi^T \lambda \delta_{\phi'} \delta_{\phi'}^T) \widehat{B}_\nu^T \delta_{s'}) \\
=&\textstyle\sum_{s,s'} (\delta_s^T B^T dB \delta_{s'})(\delta_s^T \widehat{B}_\mu \lambda \widehat{B}_\nu^T \delta_{s'})
\end{aligned}$$

$$\begin{aligned}
\mathcal{T}_d(B\mu, B\nu) &= \inf_{\lambda \in \Lambda(B\mu, B\nu)} \textstyle\sum_{\phi,\phi'} \delta_\phi^T \lambda \delta_{\phi'} \delta_\phi^T d \delta_{\phi'} \\
&= \inf_{\lambda \in \Lambda(B\mu, B\nu)} \textstyle\sum_{s,s'} (\delta_s^T B^T dB \delta_{s'})(\delta_s^T \widehat{B}_\mu \lambda \widehat{B}_\nu^T \delta_{s'}) \\
&\geq \inf_{\bar{\lambda} \in \Lambda(\mu, \nu)} \textstyle\sum_{s,s'} (\delta_s^T B^T dB \delta_{s'})(\delta_s^T \bar{\lambda} \delta_{s'}) = \mathcal{T}_{B^T dB}(\mu, \nu)
\end{aligned}$$

Next, we can prove by induction that $B_n^T d_n B_n = F^n(0)$. We skip the base case which is trivial. Assuming the statement holds for n, let $\phi, \phi' \in B_{n+1}$ such that $\phi(s) = 1$ and $\phi'(s') = 1$.

$$\begin{aligned}
&(B_{n+1}^T d_{n+1} B_{n+1})(s, s') = d_{n+1}(\phi, \phi') \\
&= \max_{a \in A} (|\mathcal{R}^a(s) - \mathcal{R}^a(s')| + \gamma \mathcal{T}_{d_n}(\delta_s^T \mathcal{P}^a B^T, \delta_{s'}^T \mathcal{P}^a B^T)) \\
&= \max_{a \in A} (|\mathcal{R}^a(s) - \mathcal{R}^a(s')| + \gamma \mathcal{T}_{B_n^T d_n B_n}(\delta_s^T \mathcal{P}^a, \delta_{s'}^T \mathcal{P}^a)) \\
&= \max_{a \in A} (|\mathcal{R}^a(s) - \mathcal{R}^a(s')| + \gamma \mathcal{T}_{F^n(0)}(\delta_s^T \mathcal{P}^a, \delta_{s'}^T \mathcal{P}^a)) \\
&= F^{n+1}(0)(s, s')
\end{aligned}$$

Now we have the desired result, which is:

$$d^* = \sup_n F^n(0) = \sup_n B_n^T d_n B_n \qquad \square$$

Algorithm 2 Synchronous declustering

$B_1 \leftarrow$ `partition_declust` of $\{S\}$ based on reward
while we seek a better approximation: $i = 1 \cdots \infty$ **do**
  $B_{i+1} \leftarrow \emptyset$
  for all $\phi \in B_i$ **do**
    add `partition_declust` of ϕ using transition P over B_i to B_{i+1}
  end for
end while

Based on the result of the theorem above, one can compute d^* using an iterative algorithm which generates partition refinements B_n and computes a metric over these partitions. Refining B_n to B_{n+1} can be done efficiently using Algorithm 1 (note that this algorithm is presented for the transition map only, but it is easily modifiable to decluster based on the reward function). The complexity of computing d_n given d_{n-1} depends on the following terms: $\phi^T \phi$, the size of the block corresponding to ϕ; B_ϕ, the set of blocks in B_n containing states in ϕ; B_n, the size of the partition after n iterations:
$O(\sum_{\phi \in B_{n-1}} (\phi^T \phi)|B_\phi||A| + |B_n|^2 |B_{n-1}|^2 \log |B_{n-1}||A|)$
The first term of the sum accounts for the construction of B_n and the second part of the sum accounts for the computation of the metric d_n. As n approaches ∞, the update algorithm runs in $O(|A|(|S||B_\sim| + |B_\sim|^4 \log |B_\sim|))$, which is an upper bound for the update at any step. See Algorithm 3 for a simple asynchronous version of Algorithm 2 that attempts to maintain the computational cost away from the latter upper bound. Later in the paper we will discuss heuristics for choosing the update classes $B \subset B_n$ in a way that maintains desirable convergence properties, similar to the asynchronous approach proposed in Comanici et al. (2012).

Value function approximation

In this section we will analyze approximation errors when using the derived partitions to represent the value function (or the model) as opposed to using the original state space representations. Just as the optimal Bellman operator (Equation 2), it is not hard to prove (Ferns, Panangaden, and Precup 2004) that the bisimulation metric operator is also a contraction mapping and that

$$||d^* - F^n(0)||_\infty < \gamma^n \max_a ||\mathcal{R}^a||_\infty$$

This result can be used to derive a bound for the approximation error induced when using intermediate partitions B_n. This bound is dependent on $M := \max_a ||\mathcal{R}^a||_\infty$,

$$\left|V^*_{B_n}(s) - V^*(s)\right| \leq \frac{\gamma^n M}{1-\gamma} + \frac{\gamma^{n+1} M}{(1-\gamma)^2} = \frac{\gamma^n M}{(1-\gamma)^2}$$

which is a special case of Equation 3. As far as non-optimal policies are concerned, one can provide approximation bounds on a set of polices representable by B_n. That is, if $\pi^a(\cdot) := \pi(\cdot, a) \in \mathcal{F}_S$ has the property that $span\{\pi^a : a \in A\} \subset colspan(B_n)$, then one can easily verify in Equation 1 that $\Pi_{B_n} \mathcal{R}^\pi = \mathcal{R}^\pi$ and $\Pi_{B_n} \mathcal{P}^\pi f = \mathcal{P}^\pi f,\ \forall f \in colspan(B_{n-1})$. Note that the set of all *open-loop policies* satisfy this property. Using again the fact that

the Bellman operator is a contraction mapping,

$$||V^{\pi}_{B_n} - V^{\pi}||_{\infty} = ||(T^{\pi}_{B_n})^n(0) - V^{\pi}||_{\infty} \leq \gamma^n \max_a ||\mathcal{R}^a||_{\infty}$$

Asynchronous partitioning

In this section, we present an asynchronous partition algorithm. Just as in the previous section, we generate a sequence of partitions and metric over these, with the property that the corresponding metrics can be transformed to a sequence of metrics converging to the desired Kantorovich-based fixed point bisimulation metric.

Let $\widehat{d^*} = \sup_n B^T d_n B$, where d_n are metrics over a sequence of partitions B_n, define inductively as follows:

- B_1 is a partition such that $\forall a, \mathcal{R}^a \in colspan(B_1)$.
- B_n is a partition that is built based on two (chosen) elements $\phi_{n,1}$ and $\phi_{n,2}$ in B_{n-1}. The partition B_n should satisfy the property that $\phi(s) = \phi'(s), \forall \phi \in B_n$ iff
  - $\phi(s) = \phi(s') \;\; \forall \phi \in B_{n-1}$ and
  - if $\phi_{i,1}(s) + \phi_{i,1}(s') + \phi_{i,2}(s) + \phi_{i,2}(s') = 2$,

    then $\forall a \in A, \forall \phi \in B_{n-1}, \; (\mathcal{P}^a\phi)(s) = (\mathcal{P}^a\phi)(s')$
- Define the metric d_n based on the Kantorovich metric over partition B_{n-1} with ground metric d_{n-1}. For $\phi, \phi' \in B_n$ and s, s' such that $\phi(s) = 1, \phi'(s') = 1$, we define $d_n(\phi, \phi')$ as

  if $\phi_{n,1}(s) + \phi_{n,2}(s) + \phi_{n,1}(s') + \phi_{n,2}(s') = 2$,

  $\max_a(|\mathcal{R}^a(s) - \mathcal{R}^a(s')| + \gamma \mathcal{T}_{d_{n-1}}(\delta_s^T \mathcal{P}^a B^T, \delta_{s'}^T \mathcal{P}^a B^T))$

  else $d_{n-1}(\widehat{\phi}, \widehat{\phi'})$ for $\widehat{\phi}, \widehat{\phi'} \in B_{n-1}$ with $\widehat{\phi}(s)=1, \widehat{\phi'}(s')=1$

  Note that this metric is well defined: the choice of states in s, s' with $\phi(s) = 1$ and $\phi(s')$ will not make any difference in either cases. This is guaranteed in the construction of the partition B_n, where we make sure that states for which the distance gets updated using the Kantorovich metric are mapped the same way in B_{n-1} **only if** transitions to elements of B_{n-1} are the same.

Theorem 2. *(Comanici, Panangaden, and Precup 2012) Given a sequence $\{K_i\}_{i=1}^{\infty}$ of subsets of $S \times S$ such that each pair from S is represented infinitely often (i.e., $\bigcap_{n=1}^{\infty} \bigcup_{i=n}^{\infty} K_i = S \times S$), and h_n is defined inductively as*

$$h_n(s, s') = \begin{cases} h_{n-1}(s, s') & \text{if } (s, s') \notin K_n \\ F(h_{n-1})(s, s') & \text{if } (s, s') \in K_n \end{cases},$$

then $\sup_n h_n = d^*$.

Theorem 3. *Let $\{B_n, \phi_{n,1}, \phi_{n,2}, d_n\}_{i=1}^{\infty}$ be a sequence of partitions and metrics generated under the strategy described above. If for every pair $s, s' \in S$,*

$$\sum_{i=1}^{n} \max(0, \phi_{i,1}(s) + \phi_{i,2}(s) + \phi_{i,1}(s') + \phi_{i,2}(s') - 1) \to \infty$$

then $d^* = \sup_n B_n^T d^n B_n$.

Proof. We will use Theorem 2 in Comanici et al. (2012). For this, we will show that $h_n := B_n^T d_n B_n$ can be computed using the algorithm in the paper cited. That is, at each time step, let K_n be the subset of pairs for which $\sum_{j=1,2} \phi_{i,j}(s) + \sum_{j=1,2} = 2$.
If $(s, s') \in (S \times S) \setminus K$, by definition

$$\begin{aligned} h_{n+1}(s, s') &:= (B_{n+1}^T d_{n+1} B_{n+1})(s, s') \\ &= (B_n^T d_n B_n)(s, s') = h_n(s, s') \end{aligned}$$

Now, for $(s, s') \in K$, by definition

$$\begin{aligned} &h_{n+1}(s, s') := (B_{n+1}^T d_{n+1} B_{n+1})(s, s') \\ &= \max_a(|\mathcal{R}^a(s) - \mathcal{R}^a(s')| + \gamma \mathcal{T}_{d_n}(\delta_s^T \mathcal{P}^a B^T, \delta_{s'}^T \mathcal{P}^a B^T)) \\ &= \max_a(|\mathcal{R}^a(s) - \mathcal{R}^a(s')| + \gamma \mathcal{T}_{B_n^T d_n B_n}(\delta_s^T \mathcal{P}^a, \delta_{s'}^T \mathcal{P}^a)) \\ &= \max_a(|\mathcal{R}^a(s) - \mathcal{R}^a(s')| + \gamma \mathcal{T}_{h_n}(\delta_s^T \mathcal{P}^a, \delta_{s'}^T \mathcal{P}^a)) \\ &= F(h_n)(s, s') \end{aligned}$$

Note also that

$$(s, s') \in K \text{ iff } \sum_{j=1,2} \phi_{i,j}(s) + \sum_{j=1,2} \phi_{i,j}(s') - 1 > 0$$

Therefore, we are guaranteed that every pair of states is selected infinitely often. The requirements of Theorem 2 are satisfied, so the algorithm in this section will compute the equivalent of an asynchronous bisimulation metric computation. □

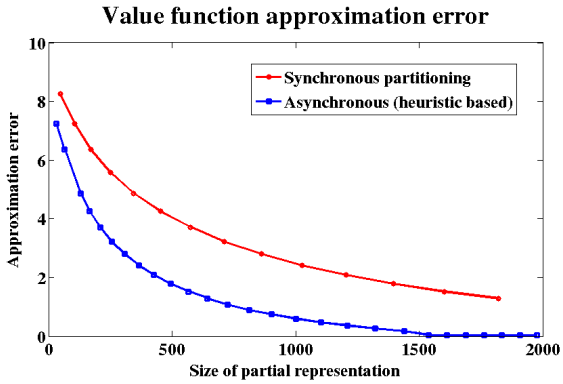

Figure 1: **Asynchronous computation:** A plot of the approximation error in the value function computation (L_∞ norm) as the size of the alternative representation is increased. This particular plot was generated on a Puddle World of size 4900.

Algorithm 3 Asynchronous declustering

$B_1 \leftarrow$ `partition_declust` of $\{S\}$ based on reward
while we seek a better approximation: $i = 1 \cdots \infty$ **do**
  Use a heuristic to select subset $B \in B_i$. Set $B_{i+1} \leftarrow B_i \setminus B$
  for all $\phi \in B$ **do**
    add `partition_declust` of ϕ using transition P over B_i to B_{i+1}
  end for
end while

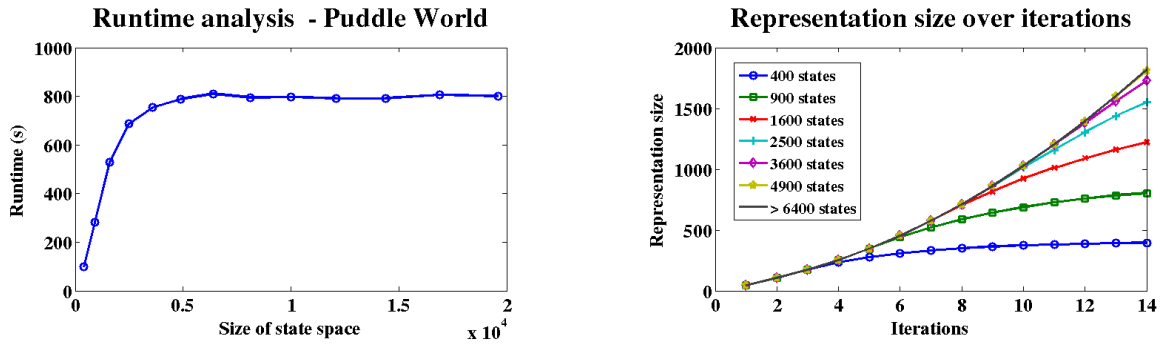

Figure 2: ***Puddle World - computing the metric.*** **Left:** A plot of the runtime as a function of state space size when computing the metric. We performed a comparison to previous work, where metrics are computed over the state space: *the runtime jumps from 129 seconds on a 400 states environment, to 1375 seconds on 1600 states*. **Right:** The number of features in the intermediate steps of the algorithm. We ran the algorithm for state spaces as large as 19600 states, but the number of features did not change substantially for sizes larger than 6400 states.

Empirical illustration

One implication of the theoretical results presented in the previous sections is that the computational complexity of computing bisimulation based representations and corresponding metrics is mostly dependent on the intrinsic complexity of the reward function and transition models. To illustrate this, we computed bisimulation metrics using the procedure presented in Algorithm 2 over a series of MDPs that increases in size, but whose structure remains the same. That is, we vary the number of states in the original MDP, but not the main task, which is to find the path from any point to a given corner by avoiding the negative rewards obtained when navigating through a puddle in the middle of the environment. More states result in a finer discretization of the original continuous puddle world problem. The well known *Puddle World* problem has a state space consisting of a grid (of varying size for the purpose of illustration); the actions available are labeled by the 4 main compass directions, and they achieve movement in the corresponding direction with probability 0.85, keep the state unchanged with probability 0.05, and move in a random direction with probability 0.1. Note that the grid has margins, which bounce back transitions that would take the agent outside the grid. The reward is dependent on the position of the state compared to a given puddle (Boyan and Moore 1995).

The left panel of Figure 2 shows the runtime of computing bisimulation metrics over partitions, for up to 14 iterations of Algorithm 2. Computation time stays roughly the same once the state space exceeds 4900 states. This is because the feature representation stays *relatively unchanged* in size and the metric computation is performed *over the set of features* and not over the entire state space. The right panel of Figure 2 shows the number of features obtained when computing bisimulation metrics using the procedure described in this paper. The key finding is that after a point, the number of features found is roughly constant even as the state space increases. This is because the complexity of the reward function and the transition system remains unchanged, and in particular, this domain is fairly simple.

To illustrate the importance of the asynchronous partition/metric update, we fixed the size of the Puddle World and we compared the value function approximation error as a function of the size of intermediate representations. In this case, for each partition we performed dynamic programming to compute the value function. In the asynchronous algorithm, we used a heuristic which selects first the largest block in order to update the partition/metric. This comes from the intuition that it would be advantageous to seek a representation that is as uniform as possible, instead of having a mix of large and small partitions. As can be seen in Figure 1, the asynchronous algorithm obtains representations of better quality in much earlier stages of the iterative framework. Note that we did not list the running time, as the time spent on obtaining such representations is relatively the same. As already discussed, runtime complexity is mostly dependent on the desired representation size.

Conclusion and future work

We presented two new ways of describing bisimulation metrics from a theoretical perspective, and we used these to design novel iterative refinement algorithms. These algorithms provide substantial improvement in terms of time and memory usage, and more flexibility in terms of guiding the search for alternative state space representations for MDPs. As illustrated, the methods we propose are not nearly as sensitive to the size of the state space of the problem.

The approach presented in this paper opens the door to more specialized strategies to finding bisimulation-based MDP representations. We illustrated the advantage of using heuristic based search strategies, but the strategy we used (which attempts to keep the size of state partitions roughly the same) is very simple, and it is likely that more sophisticated approaches would work better. For example, one could try strategies similar to prioritized sweeping, which focus on areas of the state space where the metric is changing drastically. Investigating more sophisticated heuristics and applying them to larger problems is a worthwhile direction for future work.

References

Bertsekas, D. P., and Castanon, D. A. 1989. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control* 34.

Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Athena Scientific, Bellman, MA.

Boyan, J. A., and Moore, A. W. 1995. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems (NIPS) 7*, 369–376. MIT Press.

Bradtke, S. J., and Barto, A. G. 1996. Linear Least-Squares algorithms for temporal difference learning. *Machine Learning* 22(1-3):33–57.

Comanici, G., and Precup, D. 2011. Basis function discovery using spectral clustering and bisimulation metrics. In *the 25th AAAI Conference on Artificial Intelligence*.

Comanici, G., and Precup, D. 2012. Basis function discovery using spectral clustering and bisimulation metrics. In *Lecture Notes in Computer Science, Volume 7113*.

Comanici, G.; Panangaden, P.; and Precup, D. 2012. On-the-fly algorithms for bisimulation metrics. In *the 9th International Conference on Quantitative Evaluation of SysTems (QEST)*.

Desharnais, J.; Gupta, V.; Jagadeesan, R.; and Panangaden, P. 1999. Metrics for labeled Markov systems. In *Proceedings of CONCUR99*, number 1664 in Lecture Notes in Computer Science. Springer-Verlag.

Desharnais, J.; Gupta, V.; Jagadeesan, R.; and Panangaden, P. 2004. A metric for labelled Markov processes. *Theoretical Computer Science* 318(3):323–354.

Ferns, N., and Precup, D. 2014. Bisimulation metrics are optimal value functions. In *the 30th Conference on Uncertainty in Artificial Intelligence*.

Ferns, N.; Panangaden, P.; and Precup, D. 2004. Metrics for finite Markov decision precesses. In *the 20th Conference on Uncertainty in Artificial Intelligence*, 162–169.

Givan, R.; Dean, T.; and Greig, M. 2003. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence* 147(1-2):163–223.

Larsen, K. G., and Skou, A. 1991. Bisimulation through probabilistic testing. *Information and Computation* 94:1–28.

Moore, A., and Atkeson, C. 1993. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning* 13:103–130.

Parr, R.; Li, L.; Taylor, G.; Painter-Wakefield, C.; and Littman, M. L. 2008a. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 752–759.

Parr, R.; Painter-Wakefiled, H.; Li, L.; and Littman, M. L. 2008b. Analyzing feature generation for value function approximation. In *International Conference on Machine Learning (ICML)*, 737–744.

Ravindran, B., and Barto, A. G. 2002. Model minimization in hierarchical reinforcement learning. In *the 5th International Symposium on Abstraction, Reformulation and Approximation (SARA)*, 196–211.

Sutton, R. S. 1988. Learning to Predict by the Methods of Temporal Differences. *Machine Learning* 3(1):9–44.

Yu, H., and Bertsekas, D. 2006. Convergence results for some temporal difference methods based on least squares. Technical report, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology.